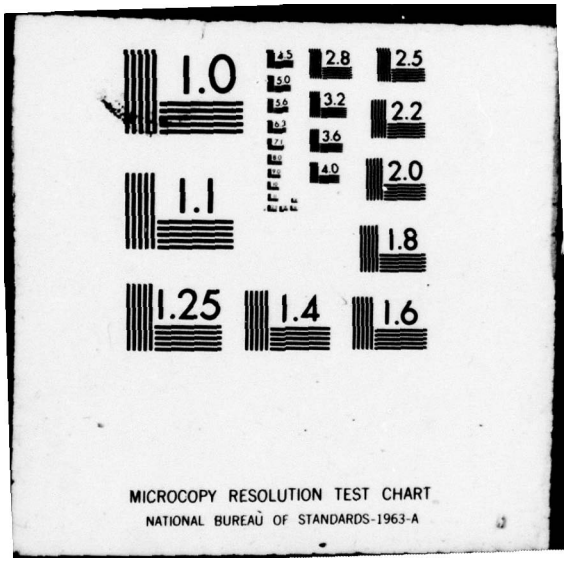
AD-A071 432 YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE F/G 5/2
SKIMMING STORIES IN REAL TIME: AN EXPERIMENT IN INTEGRATED UNDE--ETC(U)
MAY 79 G F DEJONG N00014-75-C-1111
UNCLASSIFIED RR-158 NL

1 OF 3
AD A071432


1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL II

12

AD A071432

Skimming Stories in Real Time:
An Experiment in Integrated Understanding

May 1979

Research Report #158

Gerald Francis DeJong II

DDC FILE COPY

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

This work was presented to the Graduate School of Yale University in candidacy for the degree of Doctor of Philosophy.

Skimming Stories in Real Time: An Experiment in Integrated Understanding

May 1979

Research Report #158

Gerald Francis DeJong II

This work was supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under contract N00014-75-C-1111.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1 REPORT NUMBER: #158	2 GOVT ACCESSION NO. / 3 RECIPIENT'S CATALOG NUMBER
4 TITLE (and Subtitle): (6) Skimming Stories in Real Time: An Experiment in Integrated Understanding	5 TYPE OF REPORT & PERIOD COVERED: Ph.D Thesis 6 PERFORMING ORG. REPORT NUMBER
7 AUTHOR(s): (10) Gerald Francis DeJong, II	8 CONTRACT OR GRANT NUMBER(s): (15) N00014-75-C-1111
9 PERFORMING ORGANIZATION NAME AND ADDRESS: Yale University-Department of Computer Science, 10 Hillhouse Avenue, New Haven, Connecticut 06520	10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11 CONTROLLING OFFICE NAME AND ADDRESS: Advanced Research Projects Agency, 1400 Wilson Boulevard, Arlington, Virginia 22209	12 REPORT DATE: (11) May 79 13 NUMBER OF PAGES: 220
14 MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Office of Naval Research, Information Systems Program, Arlington, Virginia 22217 (12) 238 p.	15 SECURITY CLASS. (of this report): unclassified 15a DECLASSIFICATION DOWNGRADING SCHEDULE

16 DISTRIBUTION STATEMENT (of this Report)

Distribution of this report is unlimited

DISTRIBUTION STATEMENT A
Approved for public release; Distribution Unlimited

17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

(14) RR-158 (9) Research rept.,

18 SUPPLEMENTARY NOTES

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

Artificial Intelligence	Parsing	Reading
Natural Language	Knowledge-based systems	Automatic summarization
Semantics	Inference	
Conceptual Dependency	Skim	

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

→ This dissertation describes a new method of automated text analysis. FRUMP (Fast Reading Understanding and Memory Program) is a working natural language processing system that has been implemented to demonstrate the viability of this new approach. The system skims news stories directly from the United Press International news wire and produces a summary of what it understands. FRUMP is able to correctly process news articles it has never before seen. ←

407 052 slt

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The process of interpreting input text words can be greatly simplified if it is viewed as one component of a highly integrated understanding process. In FRUMP the text analyzer is embedded in a predictive understander. This embedded in a predictive understander. This embedding is the key to FRUMP's robustness. FRUMP's integration makes all of the world knowledge and top-down predictions of the understander available to the text analyzer. FRUMP uses a data construct called a sketchy script to store its world knowledge. There is one sketchy script for each real world "situation" FRUMP knows about. The system uses this knowledge to make predictions about what might happen next in a given situation. FRUMP continually jumps to conclusions about what the text means and generates predictions about what might occur next. The text analysis process than is reduced to finding a reading of the text that satisfies these predictions. The process of looking for readings in the text is much simpler than the process of generating a conceptual structure from an arbitrary input. Thus there is no need in FRUMP for an extremely powerful English parser.

Given a new input FRUMP must be able to decide which of its sketchy scripts contains the knowledge needed to process the input. This is the process of "script selection" which is a major problem for an approach such as FRUMP's. A workable solution to the script selection problem must be computationally manageable. The process must not be significantly slowed down by the addition of more world knowledge in the form of more sketchy scripts. That is, the computational complexity of script selection must not depend significantly on the number of scripts in the system. Furthermore, the script selection process must often be completely bottom-up; most news stories cannot be anticipated before they are seen. Yet during this process, the FRUMP text analyzer must still be supplied with adequate top-down guidance. This problem is addressed and a general solution for FRUMP's purposes is given.

-- OFFICIAL DISTIRUBTION LIST --

Defense Documentation Center — 12 copies
Cameron Station
Alexandria, Virginia 22314

Office of Naval Research — 2 copies
Information Systems Program
Code 437
Arlington, Virginia 22217

Advanced Research Projects Agency — 3 copies
Cybernetics Technology Office
1400 Wilson Boulevard
Arlington, Virginia 22209

Office of Naval Research — 1 copy
Branch Office - Boston
495 Summer Street
Boston, Massachusetts 02210

Office of Naval Research — 1 copy
Branch Office - Chicago
536 South Clark Street
Chicago, Illinois 60615

Office of Naval Research — 1 copy
Branch Office - Pasadena
1030 East Green Street
Pasadena, California 91106

Mr. Steven Wong — 1 copy
Administrative Contracting Officer
New York Area Office
715 Broadway - 5th Floor
New York, New York 10003

Naval Research Laboratory — 6 copies
Technical Information Division
Code 2627
Washington, D.C. 20375

Dr. A.L. Slafkosky — 1 copy
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380

Office of Naval Research — 1 copy
Code 455
Arlington, Virginia 22217

Office of Naval Research — 1 copy
Code 458
Arlington, Virginia 22217

Naval Electronics Laboratory Center — 1 copy
Advanced Software Technology Division
Code 5200
San Diego, California 92152

Mr. E.H. Gleissner — 1 copy
Naval Ship Research and Development
Computation and Mathematics Department
Bethesda, Maryland 20084

Captain Grace M. Hopper — 1 copy
NAICOM/MIS Planning Board
Office of the Chief of Naval Operations
Washington, D.C. 20350

Mr. Kin B. Thompson — 1 copy
Technical Director
Information Systems Division
OP-91T
Office of the Chief of Naval Operations
Washington, D.C. 20350

Advanced Research Project Agency — 1 copy
Information Processing Techniques
1400 Wilson Boulevard
Arlington, Virginia 22209

Professor Omar Wing — 1 copy
Columbia University in the City of New York
Department of Electrical Engineering and
Computer Science
New York, New York 10027

Office of Naval Research — 1 copy
Assistant Chief for Technology
Code 200
Arlington, Virginia 22217

Captain Richard L. Martin, USN — 1 copy
Commanding Officer
USS Francis Marion (LPA-249)
FPO New York 09501

Major J.P. Pennell — 1 copy
Headquarters, Marine Corp.
(Attn: Code CCA-40)
Washington, D.C. 20380

ABSTRACT

Skimming Stories in Real Time:
An Experiment in Integrated Understanding

Gerald Francis DeJong II

This dissertation describes a new method of automated text analysis. FRUMP (Fast Reading Understanding and Memory Program) is a working natural language processing system that has been implemented to demonstrate the viability of this new approach. The system skims news stories directly from the United Press International news wire and produces a summary of what it understands. FRUMP is able to correctly process news articles it has never before seen.

The process of interpreting input text words can be greatly simplified if it is viewed as one component of a highly integrated understanding process. In FRUMP the text analyzer is embedded in a predictive understander. This embedding is the key to FRUMP's robustness. FRUMP's integration makes all of the world knowledge and top-down predictions of the understander available to the text analyzer. FRUMP uses a data construct called a sketchy script to store its world knowledge. There is one sketchy script for each real world "situation" FRUMP knows about. The system uses this knowledge to make predictions about what might happen next in a given situation. FRUMP continually jumps to conclusions about what the text means and generates predictions about what might occur next. The text analysis process then is reduced to finding a reading of the text that satisfies these predictions. The process of looking for readings in the text is much simpler than the process of generating a conceptual structure from an arbitrary input. Thus there is no need in FRUMP for an extremely powerful English parser.

Given a new input FRUMP must be able to decide which of its sketchy scripts contains the knowledge needed to process the input. This is the process of "script selection" which is a major problem for an approach such as FRUMP's. A workable solution to the script selection problem must be computationally manageable. The process must not be significantly slowed down by the addition of more world knowledge in the form of more sketchy scripts. That is, the computational complexity of script selection must not depend significantly on the number of scripts in the system. Furthermore, the script selection process must often be completely bottom-up; most news stories cannot be

anticipated before they are seen. Yet during this process, the FRUMP text analyzer must still be supplied with adequate top-down guidance. This problem is addressed and a general solution for FRUMP's purposes is given.

PREFACE

Science fiction writers have long anticipated machines that could converse fluently in natural language. The advent of the computer seemed to many to herald the realization of that dream. Yet computer analysis of natural language texts remains an elusive goal. This is not from lack of effort. Much money and effort was spent in the early days of computers on automatic translation between natural languages. The outcome was the realization that translation required understanding, and understanding was very difficult indeed for a computer.

In recent years artificial intelligence has directly addressed the problem of automated text analysis. However, very few systems have worked at all well. In the few that have, natural language analysis is used as a very specialized front end to a very specialized system. These systems are not easily extendible to other than their specialized inputs.

By far the more common system is one which explores the possibilities of natural language analysis in a toy system. These systems can correctly process few inputs other than the examples for which they were built. Some work on only one or two particular sentences. The claim, whether stated or implied, is then made that while only a preliminary version has been implemented a real working system is now just a matter of adding more vocabulary items and perhaps more world knowledge. None of these systems has ever been extended to the point of being practical. They remain signposts pointing along research directions never again followed.

This dissertation proposes yet another research direction. However, it is different from its predecessors in two ways. First, it recognizes the non-extensibility of previous systems as a theoretical rather than a practical problem and addresses that problem. Second, the "preliminary" implementation given, though still limited in many ways, can correctly process a much broader class of real world inputs than previous systems.

The extensibility problem is addressed by a radical reorganization of the natural language system. The process of interpreting words is completely integrated with the

process of "understanding" the concepts presented in the text.

FRUMP, the computer implementation of the approach, receives its input from the UPI news wire. The system quite routinely is able to "understand" completely new text inputs and construct reasonable summaries of them.

ACKNOWLEDGMENTS

One day the acid test of an natural language Ph.D. project will be whether or not it can write an acceptable dissertation for its designer. Sadly, that day has not yet arrived and so I had to struggle through myself. However, it would have been quite impossible without the help of the following people.

I am deeply indebted to my advisor, Professor Roger Schank. He remained confident of the eventual outcome of my research even when I had grave doubts. He has had a profound influence not only on my research but on my entire view of the world. His originality and imagination will long be an inspiration for me.

I also owe much to Professor Drew McDermott for encouraging me when I needed it most. He served on my committee and was instrumental, along with professor Gene Charniak, in helping me form my ideas on script selection. Professor McDermott and Professor David Barstow, who also read a draft of this dissertation, provided extremely useful comments.

Professor Wendy Lehnert served on my committee and read several drafts. Her insightful comments helped immensely in the preparation of this thesis.

Professor Jaime Carbonell, Professor Richard Cullingford, Jim Hendler, Professor Alan Perlis, Dr. Chris Riesbeck, and Mallory Selfridge read drafts. Mike Lebowitz and Ann Drinan also read chapters. Their comments helped me organize the final version.

I wish to thank Jim Hendler who did much of the FRUMP programming during the hectic period of my dissertation writing, Rod McGuire who wrote the natural language generator, and the rest of the FRUMP group: Natalie Dehn, Glenn Edelson, Bill Ferguson, Anne Hafer, Lewis Johnson, and Steven Slade. Jaime Carbonell, Mike Dyer, Anne Hafer, and Lewis Johnson also contributed to the current generator.

I would like to thank Walter Stutzman who wrote the software interface to the UPI wire and Bob Tuttle who built the interface hardware. I would also like to thank United Press International, in particular Mr. Bob Woodsum and Mr. James Buckner, for making the news wire available to

FRUMP.

Finally, I would certainly be remiss not to acknowledge the stimulating atmosphere provided by the rest of the students and faculty both in artificial intelligence and psychology here at Yale.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER 7

LIST OF ILLUSTRATIONS

CHAPTER 1

CHAPTER 3

CHAPTER 4

CHAPTER 6

CHAPTER 8

CHAPTER 1

INTRODUCTION TO FRUMP

1.1 What Is FRUMP?

FRUMP (Fast Reading Understanding and Memory Program) is a computer program that skims newspaper articles. FRUMP was designed to test a new theory of natural language processing. The emphasis in this new approach is the production of a robust system capable of processing a broad domain of input texts.

The main idea behind FRUMP is to integrate the parsing process with the rest of the understanding process. In FRUMP, the parser is not artificially separated from the inferencing process. This integration enables FRUMP to take advantage of the rich background knowledge that is necessary for both the parsing and inference processes. Thus FRUMP analyzes text using pragmatic expectations as well as syntactic and semantic ones.

FRUMP's domain is newspaper stories and, unlike most natural language systems, it routinely works on input neither it nor its programmers have ever seen before. It reads and "understands" actual text directly from the UPI news wire. When FRUMP "understands" a story, it builds a conceptual meaning representation from the input text. A natural language generation program then produces summaries from this conceptual structure in English, French, Spanish, Russian, and Chinese. FRUMP skims text rather than reading it for detail, and extracts only the most important information from a news article. It processes only what it believes to be the most important points in a story. The result is a very fast and efficient program which can easily process stories faster than they arrive from the news wire.

FRUMP is a script-based understander (Schank & Abelson [1977], Cullingford [1978]). It understands by identifying the scripts applicable to the input article and then bringing to bear all of its world knowledge about those situations.

FRUMP uses data constructs called sketchy scripts to store its knowledge about the world. Each sketchy script is the repository for the knowledge FRUMP has about what can occur in a given situation. When FRUMP realizes it is reading a story about a particular situation, it applies knowledge from the relevant sketchy script in order to predict what events are likely to occur.

We define the understanding of a text to be the creation of an unambiguous, conceptual representation of that text. The representation must be free of any surface lexical items and must include the events and causal relations implied by the text whether or not they are explicitly stated. As we will see, FRUMP's sketchy scripts play a central role in the understanding process.

1.2 Problems with Previous Systems

Most previous natural language understanding programs have been made up of at least two separate subsystems (for example, see Cullingford [1978], Parkinson et al. [1976], Riesbeck & Schank [1976], Wilks [1973], and Woods & Nash-Webber [1972]). A parser subsystem analyzes the input natural language text into some intermediate form. The intermediate representations range from surface representations such as case grammars (Simmons [1973], Heidorn [1975], and Woods & Kaplan [1971]) and annotated surface structures (Marcus [1977]) to representations involving conceptual primitives (Riesbeck [1975] and Lehnert & Burstein [1979]). An inferencer subsystem then builds a representation of the meaning of the input text. This involves incorporating the parser output into the meaning representation, inferring any missing events, and supplying the causal connections between the events.

BLOCK DIAGRAM OF A CONVENTIONAL NATURAL LANGUAGE SYSTEM

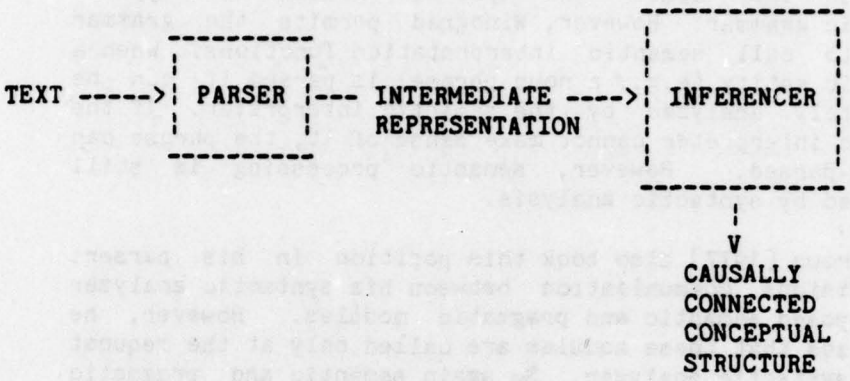


figure 1.1

Most natural language programs include various other subsystems as well such as a question answerer or a summarizer. However, these are used to demonstrate understanding and do not affect the understanding process.

Whatever the form of the intermediate representation, its purpose is always the same: to insulate the inferencer from the capriciousness of natural language. However, in all of these systems one gets the feeling that there are actually two parsers: the one everyone admits to having which produces the intermediate representation, and second one hidden in the "inferencer" which parses the output of the first parser to decide what it really means.

There is reason to believe that the separation of the parsing process from the inferencing process represents a fundamental flaw in these systems. Such separation impairs communication between the modules and places severe requirements on the parser. In FRUMP there is no intermediate representation and no parser artificially separated from the rest of the system. Instead, FRUMP's text analyzer produces representations that fit directly into its sketchy scripts. However, FRUMP is a modular system. Any reasonable sized artificial intelligence system must be modular to be comprehensible. However, modularization should not make the task at hand more difficult. The FRUMP system is broken into modules along a more natural division such that no intermediate representation is needed.

Not all previous systems disallow communication from the inferencer to the parser. Systems proposed by Marcus and Winograd permit communication both ways between the

modules.

In SHRDLU (Winograd [1972]) a systemic grammar is used to parse the input. The systemic grammar is a type of syntactic grammar. However, Winograd permits the grammar rules to call semantic interpretation functions. When a syntactic entity (e.g., a noun phrase) is parsed it can be immediately analyzed by the semantic interpreter. If the semantic interpreter cannot make sense of it, the phrase can be re-parsed. However, semantic processing is still motivated by syntactic analysis.

Marcus [1977] also took this position in his parser. He envisions communication between his syntactic analyzer and proposed semantic and pragmatic modules. However, he emphasizes that these modules are called only at the request of the syntactic analyzer. So again semantic and pragmatic processing are motivated by syntactic analysis.

Even though systems like Winograd's and Marcus's permit more communication between the parser and inferencer during processing, it is the wrong kind of communication. Semantic analysis is still done only after syntactic analysis. Semantic context is not used to provide the parser with information that can help it along. Rather it is done only at the request of the syntactic module after a certain amount of processing. Both systems defer semantic analysis until syntactic processing indicates that it is appropriate. In Winograd's SHRDLU this is done when a syntactic entity has been found; in Marcus's when his syntax analyzer reaches a choice point.

The SOPHIE system (Brown & Burton [1975]) also allows semantic and pragmatic information to influence parsing. SOPHIE uses a "semantic grammar" (Burton [1976]) to analyze natural language input. The rules in the semantic grammar try to recognize entities with certain semantic properties rather than syntactic ones. This is done by incorporating much of the world knowledge of the domain into the grammar rules. A very robust and successful system results which can handle inputs with deletions, ellipses, and anaphoric referents. However, the price paid by the semantic grammar approach is to make the language analyzer extremely domain dependent. The parser is closely tied to a very constrained micro-world. Extensive re-writing of the grammar would be necessary to change domains. While the resulting system is very impressive, the paradigm is so confining in terms of its natural language capabilities that it is questionable what the system has to say about natural language processing in general.

1.3 What is Needed to Solve These Problems?

Recently inferencers have become more and more predictive in nature. Compare, for example, the bottom up MARGIE inferencer (Rieger [1975]) to the SAM script applier (Cullingford [1978]) or consider the top down approach taken by the HEARSAY I system (Reddy et al. [1973]) It has become increasingly clear that an inferencer must know what kinds of inputs to expect in order to make sense of them. The existence of so many "frame-like" systems illustrates this (Bobrow et al. [1977], Charniak [1977], Cullingford [1978], Goldstein & Roberts [1977], Wilensky [1978]). While a frame is a very broad concept, all frame-like systems have one thing in common: they are largely top down processors. That is, to facilitate the understanding process they predict what inputs will look like before they are seen.

And yet, in natural language processing, there has been little attempt to take advantage of the inferencer's predictions when parsing natural language text. The conventional design of natural language systems makes effective communication between the inferencer and the parser extremely difficult.

To illustrate how inferencer predictions can be helpful in parsing consider the following example. Suppose the system has processed the following two sentences:

> John had a copy of Crime and Punishment.
> Bill wanted the book very much.

At this point a predictive understander such as Wilensky's PAM [1978] will have formulated the expectation that Bill will try to acquire the book. Now suppose that the next input sentence is "He took it." If the parser has been told about the prediction, it might be made to interpret the sentence correctly by recognizing that there is a meaning of the sentence that matches an outstanding prediction. However, if the parser must interpret the sentence with no help from the inferencer's predictions the task is impossible. Depending on the context, the sentence "He took it" can have many different meanings:

> Mary told John it was time for his medicine. He took it.
>
> The batter prepared for the pitch. He took it (low and outside).
>
> John saw that Bill's bishop was en prise. He took it.

Bill gave John some very sound advice. He took it.

Thus communication between the parser and inferencer facilitates the processing of each new input sentence. Without that communication, the sentence cannot be semantically interpreted by the parser alone.

In systems with syntactically oriented front ends, the problem does not arise at this point in the processing. Syntactically, the sentence "He took it" is unambiguous. It is the meaning that is unclear. Their solution is to produce only a syntactic parse and let some later semantic process assign the meaning. However, these systems suffer when semantics is necessary to prefer a syntactic parse as is often the case for prepositional phrase attachment. The sentence "Bill hit the boy with a broken leg" is ambiguous on both the syntactic and semantic levels: the boy could have a broken leg or Bill could be using a broken leg as a club. The former reading is clearly preferred to the latter on semantic grounds. However, a purely syntactic parser does not have access to this information and therefore cannot eliminate the need to produce other syntactic parses.

Whether the parser produces a meaning representation or a simple syntactic parse tree, the process often requires information available only at the semantic or pragmatic level. The solution to the problems of previous systems, then, is to make this higher level knowledge available to the parser.

1.4 FRUMP Overview

The FRUMP system has two main modules. However, it is not divided in the normal way into a parser and inferencer. One module makes predictions about what might happen next and the second module finds evidence for these predictions and fleshes them out. These modules are in constant communication with each other.

The first module, the PREDICTOR, predicts conceptual constraints on what might happen next. It does not predict individual words but rather conceptual items which may be realized in the text in any one of many different wordings. Entire conceptualizations as well as small pieces of a conceptualization can be predicted.

The second module, the SUBSTANTIATOR, tries to verify the predictions made by the PREDICTOR. Verification can be done either by finding a text input that matches a

prediction, or by deriving the prediction from what has already been understood via an inference routine. This module is called the SUBSTANTIATOR because its job is to find evidence which gives substance to the predictions.

BLOCK DIAGRAM OF FRUMP

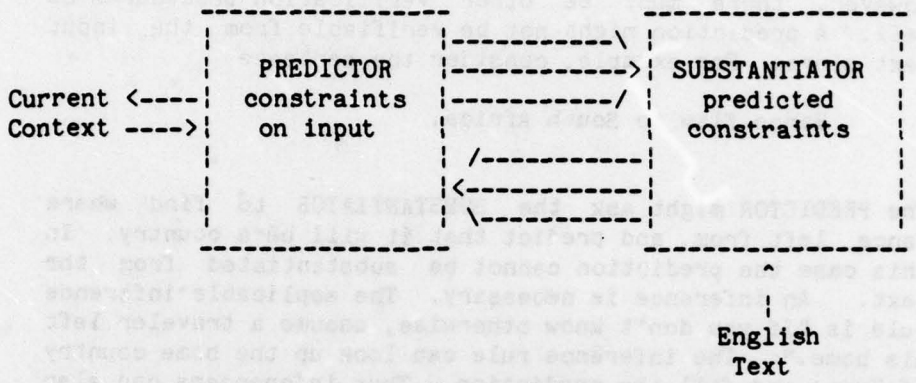


figure 1.2

The conceptual structure built by the PREDICTOR is called the current context.

During normal processing, FRUMP's text analysis is driven by what the PREDICTOR anticipates rather than by the input text. There is no conventional parser which produces conceptualizations when it is presented with an input sentence. Rather, the input text is analyzed only when the PREDICTOR wants a specific piece of information and the SUBSTANTIATOR has decided that the missing information might be found in the text. The text is only examined a little bit at a time, and then only to verify specific predictions. As the text is analyzed, the PREDICTOR revises its predictions. Thus the SUBSTANTIATOR always operates in the most complete and constrained context that the PREDICTOR can provide. There is no parsing process which must form a complete representation for a text sentence at once without help.

The predictions that drive the SUBSTANTIATOR are derived from FRUMP's sketchy scripts. Initially, of course, there will be no sketchy script. That is, when FRUMP begins reading a new article, there will be no current context. Thus, the problem of script selection presents special problems. The way FRUMP's processing gets started will be discussed at length in chapter 3.

1.4.1 The Structure within the SUBSTANTIATOR

Based on what has been understood, the PREDICTOR makes predictions about what might happen next. The SUBSTANTIATOR is then asked to verify these predictions. There are several methods that can be used in verifying predictions. One method of verifying a prediction is to look at the text. However, there must be other verification procedures as well. A prediction might not be verifiable from the input text alone. For example, consider the sentence

Vance flew to South Africa.

The PREDICTOR might ask the SUBSTANTIATOR to find where Vance left from, and predict that it will be a country. In this case the prediction cannot be substantiated from the text. An inference is necessary. The applicable inference rule is "If you don't know otherwise, assume a traveler left his home." The inference rule can look up the home country of Vance and fill the prediction. Thus inferencers can also serve as verification procedures. In fact from the point of view of the PREDICTOR inferencing and parsing are identical processes. The PREDICTOR cares only which predictions are verified and with what certainty. It does not need to know whether the verification was done from the examination of the text or by inference.

THE STRUCTURE OF THE SUBSTANTIATOR

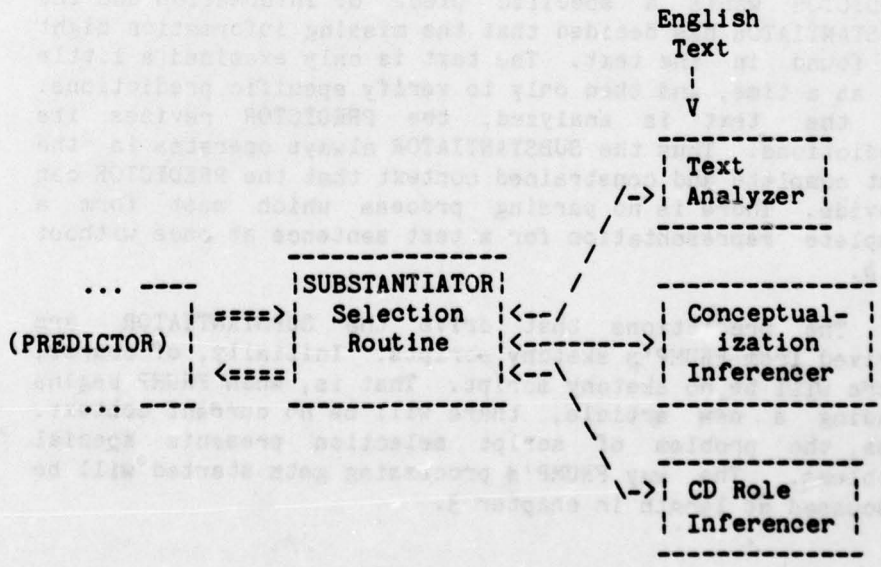


figure 1.3

Figure 1.3 shows the sub-modules that make up the SUBSTANTIATOR. The SUBSTANTIATOR has three sub-modules that actually satisfy predictions and a mechanism to select which sub-module to use for which prediction.

The conceptualization inferencer makes script related inferences about entire meaning representation. FRUMP's meaning representations are represented as conceptualizations in Schank's conceptual dependency notation (Schank [1972]). Conceptualization inferences are triggered by the other events built from the text. For example, after FRUMP processes the sentence

The U.S. broke diplomatic ties with Guinea.

the conceptualization inferencer infers that Guinea broke ties with the U.S. as well.

The CD (conceptual dependency) Role Inferencer manages the inference rules that can add a piece of a conceptualization. As discussed previously, FRUMP must be able to infer that unless contradicted, a person traveling to a location has come from his home. These kinds of inferences are organized and applied by the CD Role Inferencer. The Role inferencer has the task of efficiently locating the applicable inference rules and testing them. The major differences between the two inferencers is that the first (the conceptualization inferencer) is script based and infers an entire conceptualization at once. The second (the CD role inferencer) infers only part of a conceptualization at a time and is script independent.

The text analyzer behaves very much the same as the CD Role Inferencer. It adds requested pieces of conceptualizations. The difference is that the text analyzer adds conceptual structure by examining the text. For example, consider the following sentence:

Begin flew to the U.S.

After "Begin flew" has been processed, the PREDICTOR asks that the destination be added to the structure built. The text analyzer examines the input and fills the destination with "U.S." The major difference, as far as the selection mechanism is concerned, is that the text analyzer is more reliable. If both the Role Inferencer and Text Analyzer are able to add the same conceptual structure, it is better to use the text analyzer. An inference is a good guess based on context. However, if something is found in the text, it is almost certainly correct. Thus the text analyzer is treated as an "inferencer of first resort." Only if the text analyzer fails will the CD Role Inferencer be used.

The SUBSTANTIATOR selection routine chooses the rule that can satisfy the prediction most certainly without exceeding a cost threshold. The cost threshold is set by a parameter to FRUMP. In general the more expensive the procedure, the more certain will be the answer. Thus by setting the cost threshold low, a very fast but uncertain representation of the text is constructed. If the threshold is set higher, a more certain representation will be built but using more processing time. The cost threshold can therefore be used to control FRUMP's rate of skimming. This will be discussed in some detail in chapter 5.

1.4.2 Communication Between Modules

There are three possible outcomes from the satisfying procedure: 1) the predicted constraint might be satisfied. For example, the PREDICTOR might ask the SUBSTANTIATOR to find a country and the SUBSTANTIATOR finds "England." "England" can be considered to be a country. 2) the constraint might fail to be satisfied in such a way that indicates that the prediction is incorrect, For example, the PREDICTOR might ask the SUBSTANTIATOR to find a person and the SUBSTANTIATOR finds "England." "England" cannot be considered to be a person. 3) the constraint could simply be unsatisfied. For example, the PREDICTOR might ask the SUBSTANTIATOR to find a person and the SUBSTANTIATOR finds nothing. Each of these possibilities will now be discussed.

If the predicted constraint is satisfied, the PREDICTOR is informed of the success and modifies the current context to include the new information. This often results in a further prediction which is then treated the same way.

If evidence is found that contradicts the prediction, the PREDICTOR is also informed. The PREDICTOR then attempts to account for the failure by reinterpreting the data that led to the prediction. This can result in the modification of the current context and the formulation of other predictions.

Finally, if the constraint could not be satisfied by the selected rule but there are other rules that bid on satisfying the prediction, the other rules are tried. If no other rule has bid, the PREDICTOR is informed that the prediction was wrong. The PREDICTOR then reassesses the situation as in the previous case.

Notice once again that the PREDICTOR is not told which rule satisfied or rejected the prediction. It is not even told whether the satisfaction or rejection was due to the text analyzer or an inference. It does not matter to the

PREDICTOR; the PREDICTOR merely makes predictions and reassesses them if they fail. Thus to the PREDICTOR there is no difference between the text analyzer and the inferencers. They are both only verification routines. In practice, parsing rules are often tried first. This is because they tend to be more certain of their results. Inferences by their very nature are uncertain. However, the system treats the text analyzer as just another way to verify predictions.

1.4.3 Pragmatic Predictions Vs. Local Semantic Constraints

In a conventional modular system the parser must build a conceptual representation of the input using only syntactic and local semantic constraints. Local semantic constraints are constraints provided by word senses on how other words can relate to it. For example, one sense of the verb "jump" must have an animate agent. With an integrated system, the PREDICTOR can add global constraints. These global constraints arise from previously set up context.

To demonstrate how the PREDICTOR's global constraints can aid in the parsing process consider the following story:

> There was renewed fighting today between Israeli and Syrian forces. Syrian soldiers fired mortars at Israeli positions in the Golan Heights.

Suppose our understander is reading this story about a military engagement between two countries. In FRUMP's terms, this means the sketchy script $FIGHTING has been selected to understand the story. (Throughout this dissertation a word in capitals preceded by a "$" will indicate a script). Selecting $FIGHTING requires (among other things) identifying the countries or factions involved. The current context therefore includes the identities of the countries. In reading the third sentence, the word "fired" is encountered. The word "fire," even when known to be a transitive verb, has several meanings. It might mean "shoot" but it might also mean "put pottery into a kiln" or "terminate employment". Furthermore, the meaning cannot be decided from syntactic cues alone: "Syrians fired mortars at Israeli positions," "Bill fired clay pots at 300 degrees," and "The boss fired John at three this afternoon" all are syntactically very similar.

In the context of a story about a battle, the word "fire" ought to be interpreted as "shoot." The PREDICTOR predicts that there will be acts of shooting in the text. "Fire" is interpreted as "shoot" because in doing so a prediction is verified. Words are interpreted so as to fulfill these expectations.

In addition, once the correct word sense of "fire" is selected, the PREDICTOR can still make predictions useful to text analysis. The "shoot" sense of "fire" requires a person as the instigator of the shooting and will be satisfied with any physical object as the thing being shot at. These are constraints that the text analyzer itself can generate. However, they are very loose constraints; they must be loose for this sense of "fire" to handle sentences like

John fired his 22 at the tin can.

as well as

> Syrians fired on Israeli positions in the Golan Heights.

In fact, to be really general, the constraints of the instigator of the shooting ought to be relaxed even further. A chimpanzee, for example, might be taught to fire a gun. Thus the constraint on the agent of the shooting ought to be reduced from a person to a higher animate. This means that only a very weak prediction can be made from the "shoot" word sense of "fire."

While there are constraints made by the word sense on what can occur in the rest of the conceptualization, they are very general and of limited use. The system would do much better by allowing the understander to communicate its more constrained predictions to help in interpreting the text.

Once "fired" has been resolved to "shoot," the PREDICTOR can predict that the instigators of the shooting will be military personnel of one or the other of the warring countries, or one of their allies. This prediction can be made because this information is part of the sketchy script used to process stories about countries fighting. This constraint is much more specific than just requiring a higher animate. The PREDICTOR can also predict that the thing being shot at will be a possession, and very likely a military one, of the warring country not doing the shooting. In processing the word "fired" in the above story, the system can predict that either Israel or Syria will fill the

ACTOR slot of the conceptualization that represents the shooting.

This is a much tighter prediction than the one available from the word sense of "fired." A more constrained prediction means the text analyzer will not have to do very much to interpret the text. An expectation for the correct interpretation has already been established. These detailed predictions are based on world knowledge. In FRUMP the PREDICTOR's knowledge of how the world behaves is made available to the SUBSTANTIATOR in the form of its predictions.

The selected word sense of "fire" contains as part of its definition that in its transitive form the syntactic subject will be the agent. In Conceptual Dependency terms this is the ACTOR role. Thus the SUBSTANTIATOR looks in the text where it expects to find the syntactic subject of "fired" and try to interpret the text it finds there as a reference to Israel, Syria, or one of their allies.

1.4.4 The FRUMP Method Compared to Generate and Test

At this point the reader might be under the impression that FRUMP uses the "generate and test" paradigm (Newell [1973]). Indeed Prediction and Substantiation bears some resemblance to "generate and test." In both instances, a prediction is made which is used to motivate further processing. The difference is that in "generate and test" the precise solution is generated. In FRUMP the predictions are only constraints on what might be found. They are seldom precise specifications about what will be found. In FRUMP there is no concept of "testing." The predictions are not evaluated to a binary "yes, it works" or "no, it doesn't work." Rather the predictions are used to guide interpretation. The text analyzer prefers meanings of ambiguous words and phrases that satisfy predicted constraints. In "generate and test" the test is not allowed to modify the generated hypothesis. In FRUMP, however, the SUBSTANTIATOR fleshes out and gives substance to the predictions.

1.5 Benefits Derived from Sketchy Scripts

FRUMP's sketchy scripts help the text understanding process in two ways. First, they guide the inference process by eliminating irrelevant inferences. Only those inferences which are consistent with the script predictions are made. Second, top down script predictions can help in

correctly interpreting ambiguous words and phrases. Word meanings are selected which are closest in meaning to the predictions. These two aids to understanding are discussed further in the next two sub-sections.

1.5.1 Constraining Inferences

To illustrate the problem of constraining inferences, consider the following text fragments:

1) John carefully aimed his knife at Mary and let it fly. Blood gushed to the floor.

2) John threw the knife at Mary with all his might. Blood gushed to the floor.

Both of these fragments require the reader to supply a missing event. To account for the second sentence in each case we must assume that the knife hit Mary. Without that missing event, the other events have no apparent causal connection. Producing inferences is essential if a causally connected representation is to be built. The process of supplying data which though missing is implied by a text is called inference generation.

Inference generation is an extremely difficult process. It can be simplified somewhat by representing events in a language free form. If the inference rules are triggered by English words, then there must be separate rules of inference in the two fragments above. One rule must be triggered by the phrase "let it (the knife) fly" and the other by "threw the knife." In both cases the rules must infer that the knife will hit something. However, if the inference rules are stated in terms of the meaning of the sentences instead of actual English words then the inferencer does not need separate rules for "let the knife fly" and "throw the knife." Instead it can have one rule that if an object is moving towards another, they might collide. This one rule can then be applied to a far more general class of events which, incidentally, includes any English paraphrase of "John threw a knife at Mary." The rule is triggered by the language free meaning of the sentence, and all paraphrases must by definition have a very similar meaning.

Even with rules based on meaning, however, inferencing is a difficult process. In any situation there may be many inference rules that apply. Ensuring that the system makes the correct inferences while avoiding the irrelevant ones is

a difficult task in any natural language system.

Rieger [1975] originally postulated a general inference mechanism that generated results and reasons for each input concept. His inferencer then made a second set of inferences from the first, a third from the second, and so on. Each new set of inferences was fed back into the inferencer in order to make the new inferences. Around each input conceptualization there was an expanding sphere of inferences. Eventually the spheres in the inference space about any two inputs might intersect. That is, a common inference might be generated from the two different inputs. Only then were the relations between the original inputs understood, for only then was there an intervening set of events and causations that could lead logically from one input to the next. Rieger termed this process of connecting input concepts "knitting".

The problem with Rieger's solution is that the inferencer must make many irrelevant inferences. Consider again one of the above text fragments:

> John threw the knife at Mary with all his might. Blood gushed to the floor.

To connect these two sentences a Rieger-like inference process must have the following rules: 1) an object thrown at a target can come in physical contact with the target and apply a force to it; 2) When an object applies a force to another and is free to move, depending on the properties of the two objects either one or both can shatter, one can penetrate the other, there can be an elastic collision, etc.; 3) When living flesh is penetrated by a foreign object a wound is formed which often bleeds; 4) Unsupported objects fall toward the ground.

From these rules, the inferencer can guess at the intervening events beginning with the knife coming into contact with Mary. The problem is that there is no way to assure that only the inferences required for this causal connection are made. Rieger's method was to generate as many plausible inferences as possible. This would include irrelevant facts such as "John no longer has the knife," which would in turn produce a whole chain of other irrelevant facts. There would also probably be an inference rule such as: when someone expends energy, he becomes tired. The inferencer might follow this line to eventually predict that John will go to sleep, which has little to do with our original input. There are many such plausible inference lines the system could follow which have no chance of connecting up the inputs. However in Rieger's scheme, inferences are not assumed to be irrelevant until some other

inference line connects the inputs by knitting. It would seem that this unconstrained inferencing is combinatorially untenable.

The SAM system (Cullingford [1978]) solved the problem of unconstrained inferencing by postulating large chunks of ready-made inferences called scripts. Each script contained causally connected events that might occur in a particular situation. When the system determined that the text was about a particular situation, it made the inferences dictated by the script.

There were several problems with the SAM system. First, it did not propose a general solution to the problem of script selection. SAM worked well with a half dozen scripts, but the script selection time was linear in the number of scripts in the system. SAM would have been overwhelmed with the hundreds of scripts necessary to understand all the different situations that arise in English texts. SAM also tried to understand stories in detail. It had to account for every event in the input. This alone is not a problem, but it meant that the scripts had to be large and detailed. It also meant that if a story occurred in a way different from the way the script allowed, the story could not be understood. Thus the system was quite fragile. Finally SAM was very slow in its processing. It could take up to a quarter of an hour on a DEC KA10 to process a one paragraph story.

1.5.2 Guiding the Parser

Both Rieger's inferencer and Cullingford's script applier needed input in unambiguous conceptual form. That is, the natural language input had to be parsed before the processes could function. In both systems the inference or understanding phase was preceded by a separate parsing phase.

Rieger's inferencer was able to give the parser very little help in translating the natural language input into its conceptual representation. It was of assistance only when the parser asked a specific question about memory. Cullingford's script applier did a little better but its help was limited to instructing the parser to prefer certain word meanings while certain scripts were active. For example, in most situations the noun "check" means "bank draft." In a restaurant, though, it can also mean "charge for service." The "bank draft" meaning was the preferred meaning in the parser. However, if the input was known to be about a restaurant (i.e., the restaurant script were active) the sense that means "charge for service" was

preferred in the parser. So in the context of a restaurant, the "charge for service" sense would be tried first whenever the word "check" was encountered. When the restaurant script was activated, the script applier instructed the parser to do this.

In an ideal system (and one might imagine in the human system) the understander can predict what conceptual inputs are likely to occur in a situation (as the SAM system does). The parser ought to be able to take advantage of those predictions. Much effort in parsing is spent in selecting the correct interpretation of ambiguous words. If the parser is given no clues from the understander about what might happen next, as in the SAM system, disambiguation must be done on the basis of other words in the immediate context.

Top down predictions can be of particular help in processing pronouns. If a pronoun is encountered at a point where a specific object has been predicted, the pronoun can be resolved to be the predicted object. SAM's scripts provided the information necessary to resolve pronouns in this manner. However, due to the non-integrated approach, this knowledge was not available to the parser. In these cases the parser output contained an ambiguous referent like (HUMAN GENDER MALE). The script applier would then have to establish the correct referent later.

1.6 Situations Sketchy Scripts Can Represent

A sketchy script contains the most important information shared by all articles about a particular topic. This information includes events that are likely to occur, probable reasons for and results of those events, and constraints on the identity of and relation between characters. Of course, there are constraints on the kinds of topics that can be represented by sketchy scripts. Sketchy scripts can only represent stylized situations, (i.e., situations in which the important events happen nearly the same way in every article describing that situation). These constraints are due to the fact that all of the important events in the situation must be anticipated by the sketchy script.

To skim a story successfully a system must grasp the important facts presented. Sketchy scripts are used to guide skimming. Each sketchy script supplies the important common information about a particular topic. Thus, before FRUMP can understand a story about a particular topic, a sketchy script must be written for that topic. The sketchy script specifies what FRUMP is to look for when processing a

story about that topic.

Human readers seem to take advantage of shared information in processing texts describing stylized situations (Bransford & Franks [1971]). For example, consider the following paragraphs from the beginning of a news article.

> McKenzie, Tenn. April 30 - A businessman's family made a plea from their front porch today for the safe return of their 18 year old daughter, the apparent victim of a kidnapping.
>
> Jodie Elizabeth Gaines was last seen by her parents, Mr. and Mrs. Ben Gaines, on Friday evening when she left to spend the weekend with a cousin.
>
> Ludie Gaines said that she received a telephone call yesterday morning from a man demanding $250,000 for her daughter's safe return. There has been no word since, she said.

Once a reader has identified the topic as a stylized situation, in this case a kidnapping, he can do two things: First, he can make predictions about what will happen in the remainder of the story, and second, he can know what parts of the story are important and how they relate.

This is also the way FRUMP works. For each stylized situation, FRUMP has a sketchy script. The sketchy scripts are made up of <u>requests</u>. Each request is the conceptual representation of an expected event. FRUMP has a sketchy script for kidnappings. It contains requests which encode the following expected events:

1) The kidnappers will probably communicate a ransom demand to the family, company, or government of the person kidnapped.

2) The local police, FBI or other police agencies might be called in.

3) The ransom demand might be met.

4) If the ransom is met, predict that the kidnapped person will probably be released but might continue to be held or be killed.

5) If the ransom demand is not met, predict that the person will probably be held longer or killed but might be released.

6) The kidnappers might be apprehended.

7) If the kidnappers are caught, predict a court case trying them for kidnapping.

In addition to these facts, the reader knows that the important points in a kidnapping are the identity of the kidnappers, the identity of the kidnapped person, the identity of the group or person to whom the ransom demand was made and the nature of the ransom demand.

This is also the technique used in FRUMP. Once FRUMP has identified the situation (sketchy script) described by a text, it can read the text looking only for instances of the facts that the sketchy script predicts, much as a person might. Any part of the input text that FRUMP cannot interpret as one of its predicted events is simply ignored.

The identities of the participants in the sketchy script are <u>script variables</u>. In understanding a text, FRUMP both tries to find instances of the predicted facts and to bind the script variables to the identities given in the text.

1.7 A Sketchy Script

The following is FRUMP's sketchy script for demonstrations. It is made up of the events that are likely in a demonstration. The $DEMONSTRATION requests are:

Request 1:
The demonstrators arrive at the demonstration location.

Request 2:
The demonstrators march.

Request 3:
Police arrive on the scene.

Request 4:
The demonstrators communicate with the target of the demonstration.

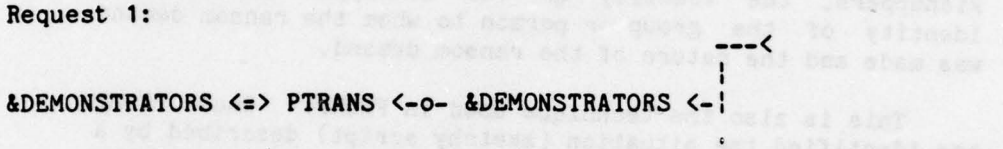
Request 5:
The demonstrators attack the target of the

demonstration.

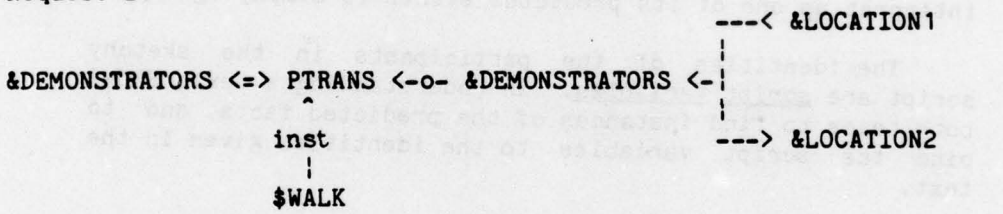
Request 6:
The demonstrators attack the the police.

In FRUMP these requests are stated in terms of conceptual dependency (Schank [1972]). Script variables appear in the requests as conceptual dependency role fillers. These are preceded by a "&". The conceptual dependency representations for the requests are:

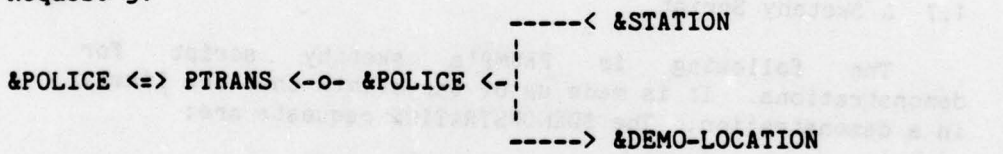
Request 1:



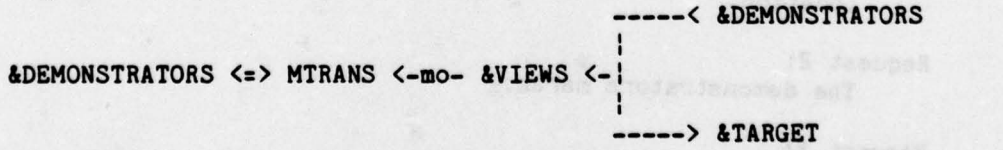
Request 2:



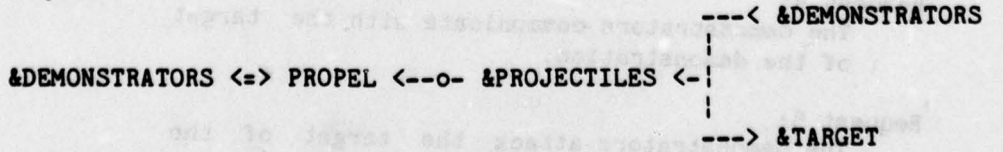
Request 3:



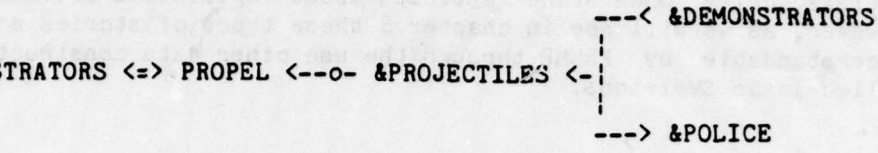
Request 4:



Request 5:



Request 6:



For each request there are also 1) semantic constraints on script variables (for example, that the demonstrators must be human); 2) constraints between script variables (for example, in the DEPORT script the country responsible for the deportation must be the same as the country the deported person leaves); 3) causation relations to other requests and references to other sketchy scripts (for example, any deaths in a vehicle accident are due to the crash event).

1.8 What Sketchy Scripts Can't Do

As was mentioned before, there are news stories for which no sketchy script can be written. These are stories for which we cannot anticipate what important events might occur. To illustrate limitations to the script approach we will now consider three types of stories FRUMP cannot handle: Variable Element Articles, Human Interest Articles, and Argumentation Articles.

1.8.1 Variable Element Articles

The first type contains stories which do describe stylized situations but where at least one important part of the situation is not predictable. We call these variable element articles. An example of a variable element article is a story about the legislature voting on a bill. Legislative proceedings are very stylized; everything happens in a prescribed order. Members endorse or denounce the bill under consideration, a vote is taken, and the bill is either ratified or killed.

Legislatures, however, concern themselves with very diverse topics. They can pass bills on anything from defense treaties to the maximum allowable number of rat hairs in a 16 ounce jar of peanut butter. It is not possible to predict what a bill is about from just the knowledge that the article describes legislative actions. And yet, the topic of the bill is the single most important point in the story. The identity of the bill's topic is the variable element in legislative

articles. Sketchy scripts alone can not be used to satisfactorily understand articles about legislative actions. However, as we will see in chapter 8 these types of stories are understandable by FRUMP through the use other data constructs called Issue Skeletons.

1.8.2 Articles that Appeal to Emotions

Articles that derive their importance from the emotional impact they have on the reader also cannot be handled by sketchy scripts. To understand why these articles do not lend themselves to sketchy script understanding we must back up for a moment.

Sketchy scripts provide a way to factor out what is common to a class of news articles. For sketchy script understanding of a situation to be successful, this common information must contain what is important in the article. If an article relies on something not part of this common information for its interestingness, sketchy scripts cannot help in it's understanding.

There are many human interest type stories that depend on emotional appeal to generate interest in the reader. For example, consider this gem which appeared in the February 16, 1978 New York Times:

> Madison, Wis. - In the life of 31 year old Hero Zzyzzx, the telephone is both travail and blessing.
>
> Mr. Zzyzzx says that he gets too many calls, at all hours, from drunks, children, insomniacs and jokers.
>
> But once in a while he gets one from "an interesting young lady," and that is why he does not obtain an unlisted number.
>
> Mr. Zzyzzx, whose name is pronounced "Zizzicks," is the last person listed in the Madison telephone book.
>
> Hero Zzyzzx is his real name, he said, a blend of Finnish, Lithuanian, Russian, French, German and central European family backgrounds. His father, Xerxes Zzyzzx, was a sailor who named his son after Hero, the man pictured on Players cigarette packs.

> Despite the hassles, Mr. Zzyzzx said there are benefits to having the last name in the telephone book. "Once in a while you get a pleasant chat with somebody," he said. "In fact, the best calls come from young ladies. I've met a number of them for drinks."

In human interest stories the important events are those that have an emotional effect on the reader. In this example the most interesting point is Mr. Zzyzzx's unique way of meeting young ladies, although the fact that Mr. Zzyzzx's fathers name is Xerxes is also particularly memorable. Perhaps it is the alliteration. At any rate, in order for a system to understand these stories it must have information about emotions of people and exactly what elicits them. This is, to say the least, very complicated and beyond what sketchy scripts were designed to handle.

1.8.3 Argumentation Articles

There is another class of articles that script processors cannot understand. These are articles that argue an issue. In an argument about an event, even an event which is part of a stylized situation, the argument often hinges on an obscure detail that only happened to occur in the particular event being discussed. FRUMP's scripts cannot help to understand these details. Sketchy scripts can only help with the important stylized information about a situation. Following an argument also often requires complete understanding of intricate cause and effect relations. Sketchy scripts, however, help in understanding only the most direct cause and effect relations, only those which are included in the stylized part of a situation.

Editorials are a good example of such articles. They often endorse a specific policy or action and try to give a persuasive argument why. To properly understand an editorial requires identifying the policy or action, deciding if the article is for or against it, and picking up the reasons given. Even when the subject of the editorial is part of a stylized situation, sketchy scripts are not flexible enough to understand the point of editorials.

For example, in a recent editorial the New York Times exhorted the New York transit workers union members to ratify a new contract. The reasons were that the city would lose tax revenues if the union went out on strike because many other commercial activities would have to slow down. They also argued that rejection by the transit workers would encourage

other unions to make greater demands to the city when their contracts expired. This, given New York's current fiscal state, is something New York could not afford, and the transit union ought to be responsible enough to realize that.

Now, of course, there could be a sketchy script for ratifying contracts. It would include knowledge such as the new monetary terms of the contract and perhaps some of the major fringe benefits, whether the union voted to accept or reject the proposal and by how much. If they voted to reject it there is a good chance they will go on strike so FRUMP should load the sketchy script for labor strikes, etc.

This is all common knowledge which is legitimate to look for in any story about contract settlements with unions. However, it has nothing to do with following the arguments in the editorial. The reasons given in the editorial are well beyond the scope of this common information.

The argument about setting a bad precedent and appealing to the union member's sense of responsibility is not particular to labor disputes. This is an arguing technique of branding the opponent with an unflattering label (irresponsible) and hoping his better judgement will accept the label and recant. There are complex questions here that have to do with individuals and organizations thinking they deserve at least as much as their peers and whether or not that is irresponsibly selfish. This knowledge is concerned with motivations and emotions of people. It is not characteristic of labor disputes alone and therefore does not belong in the labor dispute sketchy script.

The most convincing arguments are nearly always novel. People are seldom convinced by rehashing old information. Rather they must be shown an undesirable ramification not previously anticipated.

1.9 Examples

These examples show some of FRUMP's capabilities. FRUMP is a fully implemented system. The same version of the system can process all of the examples shown here and in the remainder of this dissertation.

The following story was taken from the New York Times. It demonstrates FRUMP's ability to understand the main thrust of a news story while ignoring the less important details.

INPUT:

WASHINGTON, MARCH 15 -THE STATE DEPARTMENT ANNOUNCED TODAY THE SUSPENSION OF DIPLOMATIC RELATIONS WITH EQUATORIAL GUINEA. THE ANNOUNCEMENT CAME FIVE DAYS AFTER THE DEPARTMENT RECEIVED A MESSAGE FROM THE FOREIGN MINISTER OF THE WEST AFRICAN COUNTRY SAYING THAT HIS GOVERNMENT HAD DECLARED TWO UNITED STATES DIPLOMATS PERSONA NON GRATA.

THE TWO ARE AMBASSADOR HERBERT J. SPIRO AND CONSUL WILLIAM C. MITHOEFER JR., BOTH STATIONED IN NEIGHBORING CAMEROON BUT ALSO ACCREDITED TO EQUATORIAL GUINEA.

ROBERT L. FUNSETH, STATE DEPARTMENT SPOKESMAN, SAID MR. SPIRO AND MR. MITHOEFER SPENT FIVE DAYS IN EQUATORIAL GUINEA EARLIER THIS MONTH AND WERE GIVEN "A WARM RECEPTION."

BUT AT THE CONCLUSION OF THEIR VISIT, MR. FUNSETH SAID, EQUATORIAL GUINEA'S ACTING CHIEF OF PROTOCOL HANDED THEM A FIVE-PAGE LETTER THAT CAST "UNWARRANTED AND INSULTING SLURS" ON BOTH DIPLOMATS.

SELECTED SKETCHY SCRIPT $BREAK-RELATIONS

CPU TIME FOR UNDERSTANDING = 2515 MILLISECONDS

ENGLISH SUMMARY:
THE US STATE DEPARTMENT AND GUINEA HAVE BROKEN DIPLOMATIC RELATIONS.

FRENCH SUMMARY:
LE DEPARTEMENT D'ETAT DES ETATS-UNIS ET LA GUINEE ONT COUPE LEURS RELATIONS DIPLOMATIQUES.

CHINESE SUMMARY:
MEEIGWO GWOWUHYUANN GEN JIINAHYAH DUANNJYUELE WAYJIAU GUANSHIH.

SPANISH SUMMARY:
EL DEPARTAMENTO DE RELACIONES EXTERIORES DE LOS EE UU Y GUINEA CORTARON SUS RELACIONES DIPLOMATICAS.

This story is particularly short and so took less than three CPU seconds to process. FRUMP understood that the diplomatic link from the U. S. to Guinea was ended, and it inferred that the link from Guinea to the U. S. was ended as well. The result of processing the article is a conceptual representation. Because the meaning representation is language free, it is as easy to generate other natural languages as English.

INPUT:

MOUNT VERNON, ILL, (UPI) - A SMALL EARTHQUAKE SHOOK SEVERAL SOUTHERN ILLINOIS COUNTIES MONDAY NIGHT, THE NATIONAL EARTHQUAKE INFORMATION SERVICE IN GOLDEN, COLO., REPORTED.

SPOKESMAN DON FINLEY SAID THE QUAKE MEASURED 3.2 ON THE RICHTER SCALE, "PROBABLY NOT ENOUGH TO DO ANY DAMAGE OR CAUSE ANY INJURIES." THE QUAKE OCCURRED ABOUT 7:48 P.M. CST AND WAS CENTERED ABOUT 30 MILES EAST OF MOUNT VERNON, FINLEY SAID. IT WAS FELT IN RICHLAND, CLAY, JASPER, EFFINGTON AND MARION COUNTIES.

SMALL EARTHQUAKES ARE COMMON IN THE AREA, FINLEY SAID.

SELECTED SKETCHY SCRIPT $EARTHQUAKE

CPU TIME FOR UNDERSTANDING = 3040 MILLISECONDS

ENGLISH SUMMARY:
THERE WAS AN EARTHQUAKE IN ILLINOIS WITH A 3.1999 RICHTER SCALE READING.

This story took just over three CPU seconds to process. This story illustrates why the entire story must be skimmed. The structure of news articles is such that often a FRUMP-like summary can be produced by simply parroting back the first sentence. This story illustrates why that is not always acceptable. Here the information about the strength of the earthquake would be lost. News articles are often written in a style different from most other texts. There has been some work done in classifying these styles (Eisenstadt [1976]). FRUMP was designed not as a news report processor but as a general text processor whose domain happened to be news reports. To demonstrate the general applicability of FRUMP's understanding the program does not rely heavily on knowledge about the structure of news articles not shared by other texts. The only aspect of FRUMP's processing that is at all dependent on the structure of the input text is in identifying the initial sketchy script. This identification must be done in the first paragraph. Due to the style of most news articles, this is not a serious constraint. The selection process is the subject of chapter 3.

INPUT:

THE CHILEAN GOVERNMENT HAS SEIZED OPERATIONAL AND FINANCIAL CONTROL OF THE U. S. INTEREST IN THE EL TENIENTE MINING COMPANY, ONE OF THE THREE BIG COPPER ENTERPRISES HERE. WHEN THE KENNECOTT COPPER COMPANY, THE OWNERS, SOLD A 51 PER CENT INTEREST IN THE COMPANY TO THE CHILEAN STATE COPPER CORPORATION IN 1967 IT RETAINED A CONTRACT TO MANAGE THE MINE. ROBERT HALDEMAN, EXECUTIVE VICE PRESIDENT OF EL TENIENTE, SAID THE CONTRACT HAD BEEN "IMPAIRED" BY THE LATEST GOVERNMENT ACTION. AFTER A MEETING WITH COMPANY OFFICIALS AT THE MINE SITE NEAR HERE, HOWEVER, HE SAID THAT HE HAD INSTRUCTED THEM TO COOPERATE WITH EIGHT ADMINISTRATORS THAT THE CHILEAN GOVERNMENT HAD APPOINTED TO CONTROL ALL ASPECTS OF THE COMPANY'S OPERATIONS.

SELECTED SKETCHY SCRIPT $NATIONALIZE

CPU TIME FOR UNDERSTANDING = 3457 MILLISECONDS

ENGLISH SUMMARY:
CHILE HAS NATIONALIZED AN AMERICAN MINE.

This story illustrates FRUMP's ability to identify a sketchy script. The system does not rely on "key words" to select a script. Instead scripts are selected on a conceptual basis. In this story, which is about a nationalization, The $NATIONALIZE sketchy script is selected by the presence of a conceptualization representing the abstract transfer of economic control of an industry from one country to another. Thus FRUMP does not require a semantically rich "key word" like "nationalize" to select the nationalization script. Any English paraphrase of the nationalization conceptualization will do. Here "seized operational and financial control" builds a conceptualization representing abstract transfer of economic control. The script selection process will be discussed in detail in chapter 3.

1.10 Conclusion

Once FRUMP knows that it is reading a story describing a situation for which it has a sketchy script, it can retrieve the relevant predictions by loading in the corresponding script. These predictions will then be used to help to understand the text.

Integrating understander predictions with parsing enables FRUMP to be an efficient and robust system. FRUMP is a system that understands input stories it has not been tuned for. It

processes an average story in typically 10 to 20 seconds of CPU time on a Digital Equipment Corporation PDP 20/50. Provided FRUMP has a well written sketchy script for an article it has an 80% to 90% chance of correctly picking up information from the story, and it often understands everything important from the article. FRUMP currently has 48 sketchy scripts. With these scripts it can on a typical day correctly process about 10% of the UPI wire stories. However, only about half of the UPI wire is theoretically processable with a script applier. The remainder of the stories are not scripty. Thus with 48 scripts FRUMP is achieving about a fifth of its theoretical limits. There are three main reasons for FRUMP missing the other 40% of the script UPI stories. These are, in decreasing order of importance, 1) lack of the necessary script, 2) undefined vocabulary words, and 3) an unknown or complex sentence structure used in the story.

CHAPTER 2

THE PROBLEM OF SKETCHY SCRIPT SELECTION

2.1 Introduction

When FRUMP begins reading an article, it has no context. Thus the first order of business is to establish a current context. A context is established by selecting a sketchy script to be used in understanding an article. This chapter explores some of the problems that arise in script selection and outlines FRUMP's procedures for selecting sketchy scripts.

Recall that FRUMP has one sketchy script for each news situation. The sketchy script for a situation organizes all of FRUMP's knowledge about that situation. The topic of a news article is the situation which it describes. For example, a story might describe an earthquake or the invasion of one country by another. For FRUMP to understand a news article, it must have a sketchy script corresponding to the topic of that news article. Of course, an article may refer to several situations. That is, it may have several different topics. A story describing an invasion might also mention negotiations for a cease fire. In this case FRUMP must be able to identify each of the necessary sketchy scripts. However, FRUMP initially looks for only one sketchy script to activate.

In trying to identify the initial sketchy script for an article, FRUMP confines its attention to the first paragraph. If after looking through the first paragraph FRUMP cannot choose a sketchy script, it gives up trying to understand the article. The topic of a well written news article is often given in the first sentence, and if FRUMP has not identified it in the first paragraph, the appropriate sketchy script is probably missing from FRUMP's repertoire. Rejecting the article at this point prevents

misclassifying the topic from extraneous material deep in the article.

Once FRUMP has selected a sketchy script to understand an article, that sketchy script is activated. Activating a sketchy script means that the normal events that occur in the corresponding situation will be predicted by the PREDICTOR. Thus the SUBSTANTIATOR tries to find instances in the text of the conceptualizations predicted by the currently active scripts. FRUMP uses the first paragraph to activate a sketchy script which then makes predictions about what will occur in the rest of the story.

However, the problem of selecting sketchy scripts persists throughout an article. Often articles mention several different situations. Even after an initial sketchy script is selected, FRUMP must be able to call in new sketchy scripts. For example, consider the following article:

> The New Haven Board of Education refused today to accept the agreement proposed yesterday by the teachers union. The rejection spawned an immediate citywide strike by the teachers who have been working for the past month without a contract.

The first part of this story is about labor negotiations, so that sketchy script should be activated. However, the end of the story mentions a strike. That is a different sketchy script and if FRUMP is to be able to understand the reference to a strike, it must activate the strike sketchy script as well. In this story, and many others like it, FRUMP must have several sketchy scripts active at the same time. Thus FRUMP must know how sketchy script situations normally interact and when to activate a new sketchy script, even if another sketchy script is already active.

The number of stylized situations described by news stories is very large. Perhaps as many as several hundred sketchy scripts would be required to understand all of the stylized news stories in an average newspaper. Given a story, a script based understander must choose among all the scripts it knows. Thus script selection is one of the major problems that must be overcome in building a working script-based understander. Furthermore, the solution to the script selection problem must be computationally efficient. The complexity of the selection algorithm must not depend strongly on the number of scripts in the system. Otherwise the system will be unworkable when the hundreds of scripts needed are available.

2.2 Requirements of a Solution

A valid solution to the script selection problem must have several characteristics. Before discussing FRUMP's or various other approaches to solving the problem, we will explore these required characteristics. This will help in recognizing the strengths and limitations of FRUMP's solution as well as explaining why solutions used by other systems are insufficient for FRUMP.

2.2.1 Script Selection Cannot Rely on Top Down Knowledge Alone

Initial sketchy script selection must be bottom up. Very few news events are predictable. The order they are sent over a news wire service is even more random. Therefore, FRUMP must not depend on having predictions about what type of story will occur next. Furthermore, we want FRUMP to be able to understand a significant part of a typical newspaper. This will require a very large number of sketchy scripts. Hence FRUMP cannot benefit as other AI programs such as HARPY (Lowerre [1976]) from predicting all possible inputs and then rejecting wrong guesses with a matching process. There are just too many possible inputs. Of course, we will not rule out the possibility of some previous story setting up a context in which other story types are expected. For example, if FRUMP is reading a story about an earthquake it could well predict that it will see facts about relief efforts. This is a different situation and so has its own sketchy script. FRUMP can, in this case, predict that the sketchy script for relief efforts will be relevant. However, FRUMP should not insist on having such predictions to correctly choose the sketchy script.

2.2.2 Time Efficiency of Selection

The selection process must be reasonably fast. The computer must be able to process stories in real time and much processing must be spent in parsing the story and instantiating the sketchy script. Only a small portion of time should therefore be devoted to script selection.

This constraint is a major problem in view of the potentially large number of sketchy scripts. If each sketchy script must be examined for applicability when a story is input, the script selection time will grow linearly with the number of sketchy scripts in the system. The complexity of the selection process should not depend strongly on the number of sketchy scripts in the system. If

this is not so, FRUMP will become bogged down when we add the several hundred sketchy scripts necessary for a truly general system.

2.2.3 Information Efficiency of Selection

One way to implement any script selection method is to have the system make two passes through the text: a script selection pass and a script application pass. In the first pass the system tries to identify the correct sketchy script. After doing so, the system backs up to the beginning of the story for the second pass. This time the system looks for the important elements of the selected sketchy script.

The problem with this method is that it throws away information. In selecting a sketchy script, some knowledge is gained about the article being examined: fragments of sentences will be parsed, certain script variables will be identified and bound, etc. For example, if the input text were "A bus struck a parked station wagon," in selecting the vehicle accident sketchy script, FRUMP would identify the script variable for the vehicle involved as the "bus" and object crashed into would be identified as the "station wagon." It is desirable to have the script selection process communicate this information to the script instantiation process. If the information is thrown away, the instantiation process will have to re-derive facts already understood about the article. The instantiation process should not have to re-bind the vehicle to "bus" and the object to "station wagon." The script instantiation process should be able to take advantage of any knowledge gained by the script selection process.

2.3 Solutions Used by Other Script-Like Systems

The problem of identifying what top down knowledge to use is not peculiar to FRUMP. It extends to any system that uses frame-like constructs (Minsky [1975]) to organize knowledge. To use the knowledge contained in a particular frame, that frame must be found and activated from among all the frames in the system. Most advocates of frame-like systems concede that there could be a very large number of such frames, and frame selection has been a major stumbling block in such systems. Yet there has been as yet no satisfactory general solution.

Minsky recognized the problem in his well known paper (Minsky [1975]) but had little to say about its solution except that when an input cannot be accounted for by an existing frame, a new frame would have to be selected largely by bottom up knowledge.

The SAM system (Schank [1975B]) is also a script based understander. In SAM each script is marked with a list of conceptualizations. These are conceptualizations which often indicate that the particular script will be relevant. For example, the conceptualizations attached to $RESTAURANT are representations that mean "John was hungry", or "John was going to a restaurant", or "John decided he would go to a restaurant", etc. When one of these inputs is seen, the script is activated. From then on, new input conceptualizations are matched against script conceptualizations.

There still exists the problem of matching the input conceptualization against the list of initiating conceptualizations. To make this matching process more efficient the SAM system maintained a "search list" of scripts. The search list contained scripts which for one reason or another had been predicted to be likely in this story.

There were two ways scripts were added to the search list. First, a currently active script could predict that certain other scripts might occur with it. For example, if the system knew it was reading a story about a visit to a museum, it would add the washroom script and the restaurant script to the search list since it knew visits to the washroom and restaurant might occur within museums. The second method added a script to the search list when an object often relevant in that script was seen. These were usually preferred script variable bindings for the script. If an ambulance was seen in the input, for example, the hospital script was added to the search list. From then on the hospital script was checked to see if it could account for new input conceptualizations.

When a new input was seen, an attempt was made to find a match for it in the currently active script. If no match was found or no current script was active, the input was matched against the initiating conceptualizations of each script in the search list. If a match was found there, the corresponding script was activated. If no match was found the new input could still activate a script. The methods of building the search list were not foolproof. It was quite possible for a script to be referenced in a story without having been implied by a previous script (method 1) and without the mention of one of its preferred script variable bindings (method 2). SAM, therefore, then had to match the

input against each of the initiating conceptualizations of of each of the scripts it knew of which were not in the search list. Thus in the worst case SAM had to examine each of its scripts to see if it should be activated.

More recently, Lehnert (unpublished) has suggested a similar method of script activation which is expanded to also allow certain references to settings or locations to activate scripts. Her method also includes a system of suppression devices which can override script activation. For example, John being located at a restaurant would activate the restaurant script because the setting of being at a restaurant is so closely linked with that script.

However, these activation methods often propose incorrect scripts. For example, a story might begin "An ambulance was stolen from St. Raphaels hospital yesterday, New Haven police have reported." This input should not activate the hospital script in spite of the explicit mention of both a hospital and an ambulance, both typical role fillers in the hospital script. In Lehnert's proposed method, suppression mechanisms would eliminate the proposed scripts.

In FRUMP these extraneous scripts are never proposed in the first place. Typical role fillers alone are not allowed to activate scripts. Rather the corresponding script should be activated only if the typical role filler is embedded in an appropriate conceptual event. That is, only if it is mentioned in one of a script's key requests.

Likewise, the story "Police interrupted a bank hold-up at New Haven's First Federal Savings and Loan this morning. The suspect, George Sebalto, fled into the McDonald's Restaurant across the street where he held 25 people hostage for over four hours." should not activate the restaurant script despite the proximity to the setting of McDonald's Restaurant.

Lehnert's system would allow such inputs to propose the restaurant script. The proposal would be prevented from actually activating the script by the suppression mechanisms. However, this seems inefficient. Furthermore, her method still requires the possibility of triggering a script from an entire event. For example, an input like "Israeli aircraft attacked an Egyptian radar installation," can only activate the $FIGHTING script as an entire event. "Egyptian radar installation" is not a setting which ought to trigger the $FIGHTING script, nor can "Israeli aircraft" alone be considered a typical role filler of that script. It is only the information that the planes were attacking a radar station which allows us to infer that they were even military planes.

It seems that the need for these other activation methods is obviated by the need to activate scripts from certain events. Settings and typical role fillers should only activate scripts in the context of certain specific events. An ambulance, for example, should only activate the hospital script if it is on its way to the scene of an accident, or is returning to the hospital with an injured person. Furthermore, triggering scripts only from entire events rather than the other schemes eliminates many of the misactivation problems that the other solutions suffer from.

In a recent paper Charniak [1978] proposed a partial solution to the frame selection problem. The basic idea behind his solution is that conceptual items will have two types of frame indices attached to them. There will be an action index under which the frames relevant to state changes involving this item will be listed, and there will be an object index which points to frames that are relevant to a description of this conceptual item.

Given the input

> Jack walked over to the phone. He had to talk to Bill.

Charniak wanted to account not only for activation of the telephoning frame but for the fact that most people tend to assume Jack is in a room, and many assume he is at home. To do this Charniak proposed putting the telephoning frame under the action index of telephone and the room frame under the object index of telephone. This corresponds to saying that a telephone is typically used for telephoning and that telephones typically are found in rooms. To get that Jack is at home, we need only mark the room frame that the default room is in a home.

Charniak admits difficulty with inputs such as

> There were tim cans and streamers tied to the car.

This unequivocally calls to mind the wedding frame but there is no single concept in the sentence whose index ought to include weddings. Charniak does not propose a definite solution to this dilemma but does suggest that it might be solved by discrimination nets under the indices instead of direct pointers to frames. "Streamers", for example, could have a set of context tests to see if they were tied to a car along with tin cans. If so, the wedding frame is appropriate. These tests, to be efficient, could be organized into discrimination nets. As Charniak points out, however, discrimination nets add to efficiency only if each test results in ruling out more than one frame.

The GUS system developed at XEROX PARC (Bobrow et al 1977) uses frames to carry on a dialogue to plan airline trips. GUS's frames are instances or prototypes. An instance frame is a prototype frame with some of the slots filled. They finesse the general problem of frame selection by only activating new frames from currently active frames. Thus they always require top down predictions about what frame to activate next.

Initially, a frame to carry on a dialogue about a trip specification is loaded. The dialogue frame, as all frames, has slots to be filled and attached procedures to be used to fill them. GUS's job is to fill the slots. The slots in frames are of two types. One type is filled with actual data and the other is filled with a pointer to another frame. If a slot is to be filled with a pointer to another frame, the prototype of that frame must be given along with the slot. GUS must eventually fill these slots with pointers to instance frames. The slot, however, must be marked with the prototype frame that should be used. The problem of selecting a prototype frame from bottom up information never arises.

The problem with the GUS method is that it is too constraining. It must always be the case that the prototype frame that is to fill a slot (for those slots that must be filled with frames) must be anticipated at the time the frame is written. This method works well for domains such as planning airplane trips where the facts that can be discussed are severely limited. It also has the desirable effect of helping to maintain the initiative in the dialogue. However, in a system such as FRUMP where it is impossible to anticipate the topic of the next story, a much more general selection algorithm is required.

David Rumelhart [1975] is also an advocate of organizing top down knowledge about situations. His word for the construct is <u>schema</u>. Schemata store knowledge about generic events. The schemata, like sketchy scripts, are used in understanding stories. Rumelhart's schemata, compared to sketchy scripts, are more detailed. Another difference is that there is no limit on how abstract an event schemata can represent. Sketchy scripts represent only well defined and rather rigid situations like an earthquake or an election. Schemata are used to represent things like "give" and "cause" as well. Rumelhart's schemata are hierarchically organized. Rumelhart does not directly address the problem of schema selection.

An obvious approach to the frame selection problem when dealing with natural language text is to tag certain words with the frames they typically describe. For example, the word "blackmail" might be tagged with the "blackmail

frame."

When a word is seen that calls up one or more frames, the system could try to fill the "frame slots" from other words in the text. In this way the representation is augmented while the system reads the story.

The problem with this method of frame activation is that there are times when a frame must be called up by a combination of words in the input rather than a single one. For example, the sentence "The car hit a tree." activates the venicle accident frame (or script or schema). But none of the words individually should be marked with vehicle accident: none of the sentences "the car was washed," "John hit Mary," or "the tree fell" should activate the earthquake script. One might propose that either "car" or "hit" or "tree" or all three are tagged with the vehicle accident frame. But as the three sentences which do not activate the vehicle accident frame are not extraordinary in any way, there are many such sentences which would mis-activate frames. Furthermore, there are many such examples that apparently require several words to activate a frame. A system that depended on this method alone would soon be swamped with irrelevant active frames.

2.4 The Three Kinds of Text Clues to an Article's Topic

An article often requires the reader to be familiar with the kind of event it describes. A reader who does not know what goes on in political conventions and why, for example, will have trouble understanding a news article about one. The reader must be able to identify the situation of such news articles before he can understand them. Well written news articles describing stylized events, therefore, always give very definite clues to what the stylized situation is.

Hints given by an article as to what situational knowledge is likely to be important in understanding can be classified into three types. For each type FRUMP has a method of selecting a sketchy script to activate. The method FRUMP uses depends on exactly how the script is referenced in the article being processed and on any information currently present about the article.

The three article fragments below illustrate the different ways an article can tell the reader that certain situational knowledge will be necessary in understanding. Each example tells the reader in a different way that the article is about to describe a "police arrest" situation. Thus the reader is informed that the information he has

about police arrests will probably be useful in further understanding.

EXAMPLES:

1) John Doe was arrested last Saturday morning after holding up the New Haven Savings Bank.

2) A man entered the New Haven Savings Bank bank about 10:00 am Saturday morning and demanded that a teller fill a shopping bag with money. According to witnesses, the suspect took the money to a parked car and drove off. He was caught only minutes later, however. John Doe is being held at the police station in lieu of $50,000.

3) Police apprehended John Doe, a suspected bank robber, in a drugstore in downtown New Haven. Doe was taken to the New Haven police station where he is being held in lieu of $50,000 bond.

The first example explicitly mentions that there is an arrest situation being described. An explicit mention tells the reader that a situation is important regardless of any previous contextual knowledge the reader may have. Thus explicit mention supplies bottom up information.

The second never says that an arrest occurred. Instead it uses the ambiguous word "caught." Yet every reader immediately interprets "caught" as "apprehended." This is because the context set up by the previous sentences indicates that a police arrest situation is likely. Here the police situation is implicitly mentioned by the context previously built up in the article. This is a top down activation method. A previous context is used to activate the arrest situation.

The third example tells the reader in a bottom up fashion that a police arrest occurred. Individual events from the police arrest situation are given in such a way that the reader is able to infer that the story is about an arrest. The reader, of course, must realize that a police arrest situation is being described to correctly understand the story. By the time he has finished processing the second sentence a reader must have inferred that an arrest did take place. This inference could only be made if the reader knows what typically goes on in police arrest situations and correctly identifies this article as an instance of one.

2.5 Overview of FRUMP's Three Sketchy Script Selection Methods

FRUMP has three sketchy script selection algorithms, one for each type of text clue. In each of the above examples FRUMP must decide to activate the script for police arrests, $ARREST. This script makes predictions about what will be seen in an article describing a police arrest. The $ARREST sketchy script contains requests for the following events:

1) Police go to where the suspect is

2) There is optional fighting between the suspect and police

3) The suspect is apprehended

4) The suspect is taken to a police station

5) The suspect is charged

6) The suspect is incarcerated or released on bond

In the first example,

1) John Doe was arrested last Saturday morning after holding up the New Haven Savings Bank.

the input mentions the arrest explicitly. This is called activation by Explicit Reference. In the second example,

2) A man entered the New Haven Savings Bank bank about 10:00 am Saturday morning and demanded that a teller fill a shopping bag with money. According to witnesses, the suspect took the money to a parked car and drove off. He was caught only minutes later, however. John Doe is being held at the police station in lieu of $50,000.

the arrest sketchy script is activated by the robbery script. The system must know that an arrest often follows a robbery, which is explicitly mentioned. This is called Implicit Reference. The third example,

3) Police apprehended John Doe, a suspected bank robber, in a drugstore in downtown New Haven. Doe was taken to the New Haven police station where he is being held in lieu of $50,000 bond.

gives only bottom up clues to the correct sketchy script. It is never stated that the arrest script should be activated, but only that certain events, which are part of the arrest script, took place. Activating a sketchy script from bottom up clues alone is called Event Induced Activation.

2.5.1 Explicit Reference Activation

The first example is the easiest. It is an example of script activation by explicit reference.

1) John Doe was arrested last Saturday morning after holding up the New Haven Savings Bank.

In an explicit reference activation there is a word or phrase which identifies the entire script that is to be activated. The English word "arrest" has as one of its word senses that it calls in the sketchy script $ARREST.

To be an explicit reference activation, there must be a word or phrase in the text that refers to the entire script. Refering to an event within a script is not an explicit script reference even if its occurrence always indicates that this script is appropriate. For example, consider the two sentences below.

1) A Chevy van collided with a school bus full of children.

2) There was an automobile accident involving a school bus full of childrer and a Chevy van.

The first sentence is not an explicit script reference. It does call in the vehicle accident script but there is no word or phrase such as "accident" which references that script as a whole. Instead, it gives one of the events that always occurs in a vehicle accident, namely the collision.

The second sentence, however, does activate the vehicle accident sketchy script by explicit reference. In that sentence, the phras "automobile accident" refers to the vehicle accident as a whole. It does not say explicitly that any events in the vehicle accident script took place, it says only that there was a vehicle accident in which the script variables were a school bus and a Chevy van.

Explicit reference activations cannot be done by key words. A script is not activated when a particular word is seen, but when a word sense is selected which has as its meaning a reference to a sketchy script. In the sentence

> There was an accident involving a car and an ambulance in downtown New Haven.

The vehicle accident sketchy script is activated because the correct word sense of "accident" in this context means $VEHICLEACCIDENT. This word sense can be decided upon because the text says that a car and an ambulance (both vehicles) were involved. In the sentence

> Billy had to change his trousers because he had an accident.

the vehicle accident sketchy script is not called up because a different sense of the word "accident" must be used. Thus individual words are not used to activate sketchy scripts but individual word senses.

2.5.2 Implicit Reference Activations

Consider the second example:

> 2) A man entered the New Haven Savings Bank bank about 10:00 am Saturday morning and demanded that a teller fill a shopping bag with money. According to witnesses, the suspect took the money to a parked car and drove off. He was caught only minutes later, however. John Doe is being held at the police station in lieu of $50,000.

In this example the sentence "He was caught only minutes later, however". is extremely ambiguous out of context. An understander must decide who "he" refers to, correctly disambiguate "caught", and infer who caught him. On the basis of this example sentence alone it is impossible to select the $ARREST sketchy script; there is simply not enough information in the sentence.

This example demonstrates activation by implicit reference. Sketchy scripts often occur in conjunction with other sketchy scripts. In this example, we know that the sketchy script $ROBBERY is often followed by $ARREST. A sketchy script is activated by implicit reference when a sketchy script that is known to often precede it is activated. In the above example, $ROBBERY is activated by

explicit reference. FRUMP's world knowledge includes the fact that robberies often lead to arrests. Since causally connected events are likely to be reported in the same news article, FRUMP activates the $ARREST sketchy script.

Activation by implicit reference is not often absolutely necessary. There is usually some other way to get around making the prediction about what script will occur next. Suppose, for example, that the sentence "he was caught only minutes later, however" were changed to "He was charged with armed robbery." Then the story could be understood by recognizing the $ARREST sketchy script in a bottom up fashion; the sketchy script is then selected using event induced activation. The event of charging someone with a crime strongly indicates that $ARREST is the appropriate sketchy script for this situation.

However, even when the sketchy script might be recognized without implicit reference, there are advantages to having implicit reference as a separate activation type. For example, the word "charge" in the amended example is ambiguous. "Charge with a crime" has a very different meaning (and therefore a different conceptual structure) than "charge with electricity" or "charge with a responsibility" or "charge" meaning to demand payment. If the sketchy script $ARREST were to be selected bottom up, the word "charge" would first have to be disambiguated; only "charge with a crime" should activate $ARREST.

With implicit reference activation the correct meaning of "charged" can be selected immediately. If there were no implicit reference activation, some process would have to realize that there are several dictionary definitions for "charge", and, on the basis of what was found in the rest of the sentence, choose one. In this case, "with armed robbery" strongly suggests that "charge with a crime" is the appropriate meaning.

If, however, the $ROBBERY sketchy script is allowed by implicit reference to activate $ARREST then $ARREST is already active when the word "charge" is input. FRUMP can use the fact that there is an active request looking for "charge with a crime" to prefer that meaning of "charge" immediately. In this case, the ambiguity of the word "charge" is not noticed.

Furthermore, if the "charge with a crime" meaning of "charge" can be immediately selected, the system knows that somewhere in the remainder of the sentence the actual crime will be specified. Thus using implicit reference in the amended example, the system gets essentially for free the disambiguation of the word "charge" and the top-down prediction of how to process the crime that will be

mentioned later in the sentence.

2.5.3 Event Induced Activation

The last example demonstrates event induced activation. Here, the sketchy script is activated by bottom up clues from the input text.

> 3) Police apprehended John Doe, a suspected bank robber, in a drugstore in downtown New Haven. Doe was taken to the New Haven police station where he is being held in lieu of $50,000 bond.

In this case, the event of the police apprehending a suspected criminal is sufficient for people to realize that knowledge about arrests (i.e. $ARREST) will be relevant. What does this mean in FRUMP terms? This means that the event of the police apprehending a suspect is central enough to the $ARREST sketchy script that when a conceptualization for that event is seen, the $ARREST sketchy script should be activated. We call the events which are central to a sketchy script the <u>key requests</u> of that sketchy script. When a conceptualization is found that is a key request of a sketchy script, that sketchy script is activated.

Thus event-induced activation involves building a conceptualization and testing whether it is a key request for any sketchy script. The key request test must be done by efficiently searching through all sketchy scripts for those in which the particular conceptualization is central to the script. FRUMP must have an efficient method for event induced activation because it occurs so often in everyday news articles. In the next chapter we will explore all three activation methods in detail.

CHAPTER 3

FRUMP'S SCRIPT SELECTION ALGORITHMS

3.1 Introduction

In this chapter FRUMP's selection algorithms will be described in some detail. A major part of the chapter is devoted to event induced activation which is the hardest and most interesting method of script selection. At first it will be assumed that entire conceptualizations will be supplied to the selection algorithms when needed. That is, it will be assumed that a powerful enough parser exists so that the selection algorithm need not concern itself with the problem of mapping the natural language input into conceptual dependency representations. This assumption is far from reasonable; no such powerful parser exists in FRUMP. In the final sections of this chapter it will be shown how this assumption can be dropped.

3.2 Explicit Reference Activation

In English there are words whose meaning cannot be captured in simple conceptual dependency representations. These are words like "accident" (as in auto accident), "strike" (as in a labor strike), and "invasion." Each of these words has a word sense which refers to an entire stylized event sequence, not an individual event, state, or state change. Thus they cannot have simple conceptual dependency representations as their dictionary definitions. FRUMP's definition of these word senses includes a reference to an entire sketchy script. When FRUMP chooses one of these word senses the text is making reference to an entire sketchy script situation. Sometimes these references can be anticipated. In these cases the current context built from the text thus far can account for the reference to the

sketchy script. If the current context cannot account for the script reference, it means this is a new topic in the article and FRUMP should activate the corresponding sketchy script. These sketchy scripts are activated by explicit reference.

What does it mean for a current context to "account for" the reference to another sketchy script? Consider the two examples below.

1) There was an automobile accident at the corner of Grove and Prospect Streets.

2) The National Safety Council released figures today indicating that the chance of dying in an automobile accident has fallen dramatically the past year.

In the first example, assuming no currently active sketchy scripts, the phrase "automobile accident" will activate the sketchy script $VEHICLE-ACCIDENT. In the second example, "National Safety Council released figures" will activate the script for government agency reports. One of the important pieces of information in such stories is the topic of the report. Thus there will be a request in the active government report script looking for the report's subject. This time when the reference to $VEHICLE-ACCIDENT is found, it will be interpreted as part of the report's subject. There is an active script that can account for the script reference and so no sketchy script is activated.

3.2.1 Mis-Activations

However, suppose FRUMP does not have a sketchy script for government agency reports. Then FRUMP will see "automobile accident" as an explicit reference to the vehicle accident sketchy script. FRUMP will then read the rest of the story looking for events it considers likely in vehicle accidents. This is a typical mis-activation of a sketchy script.

There are two reasons why this kind of mis-activation is tolerable. First, the failure to recognize the correct sketchy script (in the above example) is due to a lack of information, not a fault in FRUMP. Failures due to insufficient scripts or lack of vocabulary are not failures of the control structure of FRUMP. Second, if the wrong script is activated (in this case the vehicle accident sketchy script), very likely nothing in it will be satisfied. That is, nothing will be instantiated. If no sketchy script is instantiated by a certain article, it will

simply be ignored. Only if there happens to be something which instantiates the mis-activated script will the story be misunderstood.

3.3 Implicit Reference Activation

There are many news stories which report several different but strongly related situations. When the relationship between the various situations is such that the existence of one indicates the presence of others, the others are activated by implicit reference.

For example, if an article is found to report a flood or volcano eruption or other natural disaster, there will likely be relief aid from the Red Cross or other countries which will also be reported. These are two completely separate situations. In the disaster we expect to see one set of facts reported (like the location, casualties, estimates of damage, etc.) and in the relief efforts we expect to see another (the form of the relief aid, how much aid was extended, who it is from, etc.). Even though the two situations are separate, the existence of the flood strongly indicates that relief aid to the inundated country will also be reported.

3.3.1 Is Implicit Reference Activation Really Necessary?

It might be proposed that since the sketchy script for relief aid is usually present in stories about floods and must always be looked for in such articles, it ought to be part of the flood sketchy script. This is undesirable for two reasons. First, there are many other sketchy scripts which, like $FLOOD, imply relief efforts. If the information were to be stored in $FLOOD and not as a separate sketchy script, it would have to be duplicated in each of these other sketchy scripts as well. That is, the knowledge in the relief sketchy script would have to be included in $FLOOD, $EARTHQUAKE, $TIDALWAVE, etc. Second, there are articles which are entirely devoted to describing the relief and rescue actions following a large natural disaster. FRUMP must therefore be able to access the information about relief efforts independent of the relief effort's cause.

Given that a separate sketchy script is needed for such things, why not simply use one of the other activation procedures? Consider the following fabricated newspaper article:

> John Doe was injured when the motorcycle he was driving was run over by a semi tractor trailer. Miraculously, Doe was only slightly injured. His condition was said to be good but he will remain under observation.

The last sentence can only be accounted for by the hospital sketchy script. The "good condition" is interpreted as the condition released by the hospital and "remain under observation" means remain at the hospital under observation of doctors. FRUMP must realize this if it is to understand the article.

Perhaps, however, one of the other activation procedures can be used to activate the hospital sketchy script. There is no direct mention of hospital so $HOSPITAL cannot be activated by explicit reference. However, maybe event induced activation can help. In that case, there must be some conceptualization that the sentence builds which can activate the hospital sketchy script. It would seem that the only concepts which might activate the sketchy script are built from "good condition" and "under observation". However, without the context set up by the vehicle accident these seem insufficient. For example, consider the next article:

> With the Ali - Spinks fight approaching, many boxing experts question Ali's ability to go the distance and are watching him closely. No one in the Ali camp would officially comment on the veteran boxer's chances. His condition was said to be good but he will remain under observation.

This has the same final sentence as the previous article but certainly does not activate the hospital sketchy script. Nor is the hospital script called up and rejected. Rather it never occurs to us that the hospital sketchy script is relevant at all. The problem is that phrases like "good condition" and "remain under observation" are simply too general to be tied to a particular sketchy script.

Thus there must be a different activation method for the motorcycle accident example. The activation must be based largely on the context provided by the article. This is exactly what implicit reference activation does.

3.3.2 Issue Skeletons

Implicit reference activation and sketchy script interaction are handled by Issue Skeletons. The main function of issue skeletons is to connect related stories. However, they are also used for implicit reference activation. Issue skeletons will be discussed in detail in chapter 8. Here we will discuss them only to the extent that they are used for script activation.

An issue skeleton is a data construct that organizes events at a level higher than scripts. There are times when understanding events as separate scripts is insufficient. For example, suppose a set of articles reported a natural disaster followed by a Red Cross relief effort. The articles might be understood as instantiating two completely separate scripts. One for the disaster and one for the relief effort. However, this misses a very important fact. The relief effort was initiated to help the disaster victims. The system cannot be said to understand these articles unless it knows that the relief effort was a response to the disaster. The disaster and relief efforts are not two separate situations that have nothing to do with each other. This information causally connecting sketchy scripts is stored in an issue skeleton. Basically the natural disaster issue skeleton says that disasters are often followed by relief efforts so if a relief effort is seen in the same location and shortly after a natural disaster, assume that the relief effort was initiated to help victims of that disaster.

Anytime several news situations must be connected to be understood, they form a news issue. Issue skeletons dictate how news issues normally progress. There are many news issues each requiring an issue skeleton. A war, for example, is made up of battles, cease fires, peace talks, and appeals to allies for help all interspersed. There must be a separate sketchy script instantiated for each of these individual situations. The situations must then be tied together with an issue skeleton. Congressional action on a particular bill must be represented with an issue skeleton. The action is typically made up of a number of debates followed by a vote. As another example, a political campaign is made up of any number of campaign activities, a convention, and a general election. There are many such news issues.

Attached to each sketchy script is a list of the issue skeletons in which it can appear. $STORM, $EARTHQUAKE, $FLOOD, etc. are all marked that they can initiate the natural disaster issue skeleton. Thus when a flood sketchy script is instantiated, a new natural disaster issue skeleton is initiated. On the basis of that initiated issue

skeleton, FRUMP predicts that it might see secondary disasters like fires from broken gas mains and if the disaster is sufficiently bad there will probably be relief efforts. Since the predicted relief efforts correspond to a specific sketchy script, the relief efforts sketchy script is activated.

When implicitly referenced sketchy scripts interact via issue skeletons their respective script variables must match in certain specifiable ways. In the case of a bank robbery, for example, the issue skeleton predicts an arrest situation but it is understandable with the robbery only if the person arrested in $ARREST is the same as the thief in $ROBBERY and the crime he is charged with is robbing the bank. If these constraints are not met then FRUMP has probably misunderstood the story. This means the scripts fit together in a more complicated way than FRUMP can understand, and the implicit reference should be disregarded. To illustrate the constraints on how situations must conform consider the following two variations of a story:

> 32 year old John Doe was killed in a shooting in a downtown New Haven bar yesterday. Witnesses said he had argued with his brother Frank Doe who drew a gun and fired seven times at point blank range. New Haven police reportedly have charged Celia Baker with first degree murder.

> 32 year old John Doe was killed in a shooting in a downtown New Haven bar yesterday. Witnesses said he had argued with his brother Frank Doe who drew a gun and fired seven times at point blank range. New Haven police reportedly have charged Frank Doe with petty larceny.

These stories seem silly. The reason is that the variables of the murder and arrest scripts do not match properly. After reading about the murder we expect to see an arrest. However, we can predict who will be arrested and why. If the script variables are to be conformable, the shooting suspect must be the one arrested and he must be charged with murder. In the first story, we expect to see Frank Doe, not Celia Baker charged. The second story is peculiar because the charge is "petty larceny" instead of the expected "first degree murder." "Celia Baker" and "petty larceny" grossly violate our expectations. These expectations are important for FRUMP because they can be

used to test whether FRUMP has understood the interaction correctly. If they are not satisfied, the two sketchy scripts very likely do not fit together the way FRUMP assumed. A more powerful understander capable of hypothesizing complex causal interactions might be able to make sense of the new situation by manufacturing some bizarre context. This is well beyond the capabilities of a script applier such as FRUMP. To minimize the chance of further misunderstanding the article, the best course for FRUMP is to simply ignore the implicit reference.

In addition to constraining how interacting sketchy scripts can share variables, issue skeletons can supply default causation information. For example, consider the following news story:

> One of the worst blizzards in history paralyzed New York City yesterday. Drifts to six feet blocked roads and kept schools closed for the second day. Sub-zero temperatures, in places reaching twenty below, aggravated already serious conditions.
>
> There were scattered reports of power outages, the worst in Brooklyn where widespread looting and vandalism were reported. Mayor Koch called on off duty policemen to return to their jobs but has not yet asked the governor for National Guard assistance.

In this story, a blizzard is reported to have caused a blackout. There are separate sketchy scripts for blizzards and blackouts. The first paragraph reports facts about the blizzard and the second reports facts about the blackout. However, it is nowhere explicitly stated that the blackout was due to the blizzard. Instead it relies on the reader's knowledge that serious blizzards can cause blackouts. In FRUMP this knowledge is supplied in a general way by an issue skeleton. The issue skeleton provides the information that natural disasters might cause other disasters. $BLIZZARD is, of course, marked as a natural disaster. When the blizzard sketchy script is instantiated, a natural disaster issue skeleton is initiated. The natural disaster issue skeleton contains the fact that natural disasters can cause secondary or unnatural disasters. When "power outages" is seen, FRUMP can interpret it as a secondary disaster caused by the blizzard.

It is important to understand how the blizzard and blackout are related. That is, some interscript causal connections must be established. For this story, FRUMP can

make the causal inference via the natural disaster issue skeleton. These causal inferences can also only be made if certain constraints between the script variables in the old and new sketchy scripts are satisfied. In the case of blizzards and blackouts, there is only one constraint on the sketchy script variables: it must be the case that the location of the defective electrical power equipment is at or near the location of the blizzard (since no faulty electrical power equipment is discussed, it can be assumed to be at the location of the blackout). Only when this constraint is satisfied can the implicit causal inference that the blackout was caused by the blizzard be made and the blackout incorporated into the natural disaster issue skeleton. Again, if the constraints are not satisfied, the secondary disaster sketchy script is best ignored.

3.4 Event Induced Activation

As was pointed out before, the most common indication that a particular sketchy script should be active is the presence of an event which is central to that sketchy script. As this is the most common method, FRUMP must have an efficient algorithm to deal with it. The time of the algorithm should not increase greatly if FRUMP is given many more scripts. That is, the time complexity of the event induced activation algorithm should be less than linear in the number of sketchy scripts the system has. In this section we will first investigate some of the difficulties with bottom up script activation. Then FRUMP's algorithm will be discussed.

3.4.1 Bottom Up Problems

It is essential to be able to activate sketchy scripts in some sort of bottom up fashion. As explained before, FRUMP cannot depend on having predictions about what kind of news article is likely to occur next. Charniak [1978] in discussing frame selection has suggested that conceptual objects, states and actions have LOCATION and ACTION indices attached to them. The LOCATION index stores the frames for where the concept is likely to occur and the ACTION index stores frames in which the concept is often used. If a new conceptual input cannot be incorporated into the currently active frame, the frames under the components of the input are examined as candidates to activate.

This is sort of an advanced conceptual version of the system used by Rumelhart. However, there are concepts which must activate sketchy scripts where it would be a mistake to

mark any of the component conceptual items with the sketchy script. In a sense it is not the component concepts that activate the sketchy script but rather the particular configuration of components. For example, the conceptual representation for the sentence

The ground trembled.

contains the concept for "ground" or "earth in a local area", the concept for "motion" and a modifier specifying that the motion is "cyclic".

It would be a mistake to tag any of these component concepts with the sketchy script $EARTHQUAKE. The sentence

The ground was covered by fog.

also uses the concept of "earth in a local area" but should not bother the system with calling in the sketchy script for earthquakes. The sentence

John's hands trembled.

contains the concept for the same kind of motion but also should not suggest loading the earthquake sketchy script.

In addition there is a hidden problem if we choose to use the participants in the action to call in sketchy scripts (i.e. list $EARTHQUAKE under "earth"). The problem comes from the hierarchy of object concepts. Consider the sentence

New York City shook yesterday.

Of course, we do not want to store $EARTHQUAKE directly under the dictionary entry for New York City (because then we would have to store it under Moscow, Topeka, Sioux Falls, Eastern Turkey, Carnes, etc.). Instead $EARTHQUAKE should be stored under "ground" and each of these entries will be marked that they can be considered a "location". "Location", in turn, is marked that it can be considered as "ground". This means that in the above example, the system must retrieve the fact that New York City can be considered a "location", find that locations are "ground", and look there for proposed sketchy scripts. However, there is no reason, at this point to expect that "location" is the important property of New York. New York City can also be considered a city and so all of the sketchy scripts that are proposed by "city" must be tried. New York is also a harbor so all the sketchy scripts under "harbor" must be tried. New York is also a... The series is endless. Yet all the properties of New York City must be present because there are cases where the system must have the information that

New York City is a city and a harbor and so forth.

Of course, one could argue that component concepts should call in many sketchy scripts and that it is just part of the job of the system to weed out these irrelevant sketchy scripts. This might be implemented with consistency checks built in to each sketchy script. The earthquake script, for example, could at activation time test to see what was involved in the back and forth motion. Only if it turned out to be ground or something firmly connected to the ground would $EARTHQUAKE ultimately be added to the list of active scripts.

However, activating a sketchy script, while not a costly process, is not something that should be done thousands of times while processing a story. Furthermore, the knowledge needed to reject a sketchy script would seem to be the same knowledge needed to avoid proposing it in the first place. With a little more clever organization, the knowledge should be available at the time it is needed.

3.4.2 FRUMP's Solution

In the previous section it was argued that event induced activation depended not on single concepts present in the input (like New York) but on complex concepts made up of specific combinations of individual concepts (like ground shaking).

These complex concepts are similar to major requests from the sketchy scripts they activate. In each sketchy script there are one or more requests which, if satisfied, strongly indicate that the particular sketchy script is appropriate to use in understanding the current story. These requests are called the sketchy script's key requests. For example the key request for the earthquake sketchy script is the one that looks for a geographical location undergoing back and forth motion. Each sketchy script can be activated if one of its key requests is found.

The entire key request need not be seen. Rather only a part of it is necessary to activate the sketchy script. For example, the key request of the earthquake sketchy script specifies not only the location and the fact that it is moving but how violently it is moving, the time of the event, etc. These are things that can be present but are not necessary. In fact, the input need not specify a location but only that ground is being shaken. Thus only a simplified version of the entire request need be seen in order to activate a sketchy script.

The simplified requests from a script that can activate the script are called that script's script initiators. Script initiators are the minimum conceptual structure necessary to activate sketchy scripts.

For example, the script initiator for the earthquake sketchy script specifies in conceptual terms that a geological force is moving some ground and the motion is cyclic. Often part of a script initiator must be inferred. That a geological force is responsible for the motion must be supplied by world knowledge in the sentence "New York City shook yesterday."

The key requests of all the sketchy scripts can be collected into a long list. The list is made up of conceptual dependency representations, and associated with each is the sketchy script that it came from. The associated sketchy script is the script that should be activated if the representation is found in the input. This list is considerably shorter than the total number of requests in the system but is still very large. To select a sketchy script the system must compare each new input conceptualization to this list until a match is found. The corresponding sketchy script will then be activated to understand the story.

3.4.3 Matching Key Requests

The problem in event induced activation is finding some way of organizing these key concepts so they can be matched easily. The matching process must also be able to provide the text analyzer with top down guidance in interpreting the input.

From efficiency considerations, the complexity of the matching method must be less than linear in the number of sketchy scripts. No solution to the matching problem is viable if the complexity of the algorithm is linear, even with a small constant. FRUMP currently has 48 sketchy scripts, but it would have to contain hundreds if it were to process all articles about scripty situations. Thus the selection time cannot be allowed to increase dramatically with an increase in the number of scripts. If it does, the whole approach to script selection will become unworkable.

This leaves us with two possibilities. The key requests can be indexed by a hash coding technique, or by discrimination trees. All of the less-than-linear search techniques are variations on one or the other of these two processes.

The solution FRUMP uses is to organize these concepts into discrimination trees. Sketchy script initiator discrimination trees (SSIDTs) both make searching the concept space efficient and distribute the matching process throughout the tree. Spreading out the matching process is useful because, as we shall see, the text analyzer can then be driven by small changes in conceptual structure at the decision points.

3.4.3.1 Conceptual Dependency -

Discrimination trees were chosen over hash coding in the key request matching process because they enable the process to give top-down guidance to the text analyzer. The notion of lexical decomposition into primitives is essential for the success of FRUMP's matching algorithm. Since it is so central to the script selection procedure, a brief review of Conceptual Dependency is given here. Readers already familiar with Conceptual Dependency may wish to skip to the next section.

All of FRUMP's requests (and therefore sketchy script initiators since they are partial requests) are stated in terms of Conceptual Dependency. These conceptual structures are made up of roles and role fillers. For example the conceptual dependency structure for the English sentence

John went to New York.

is

```
(ACTOR      JOHN
 <=>        PTRANS
 OBJECT     JOHN
 TO         NEWYORK)
```

The form of a conceptual dependency structure is

(role1 role-filler1 role2 role-filler2 ...)

Thus in the above conceptual dependency structure ACTOR, <=>, OBJECT and TO are roles and JOHN, PTRANS, JOHN and NEWYORK are their respective role fillers.

Conceptual dependency structures are divided into three kinds of concepts: actions, states and state changes. Actions are denoted by the presence of the <=> role, states by the presence of the IS role and state changes by the

presence of the TOWARD role. The conceptualization above is therefore an action type conceptual dependency.

3.4.3.2 Two Sketchy Script Initiator Discrimination Trees -

FRUMP has one sketchy script initiator discrimination tree for each kind of conceptualization: there is an action SSIDT which contains all the script initiators that are conceptual dependency actions, a state SSIDT which contains all the script initiators that are states and a state change SSIDT which contains all the script initiators that are state changes. Thus if the input were the conceptualization for "John went to New York", FRUMP would look through the SSIDT for actions.

An SSIDT is an n-way branching discrimination tree. That is, each node can have any number of arcs emanating from it. At each node the filler of one conceptual role of a conceptualization is tested. Each arc leaving a decision node corresponds to a possible outcome of the test performed. Each leaf of the SSIDTs points to a sketchy script which will be activated if it is reached. SSIDTs are static structures - they are not changed during text processing.

New input conceptualizations from the text are submitted to the proper SSIDT (the action SSIDT for an action concept, the state SSIDT for a state concept etc.). If the tests performed on the submitted conceptualization eventually reach a leaf node, the sketchy script at that leaf node is activated. If at any node there is no arc which matches the role filler being tested, the conceptualization cannot lead to a leaf and therefore cannot activate a sketchy script so it is rejected.

SKETCHY SCRIPT INITIATOR DISCRIMINATION TREE FOR ACTIONS

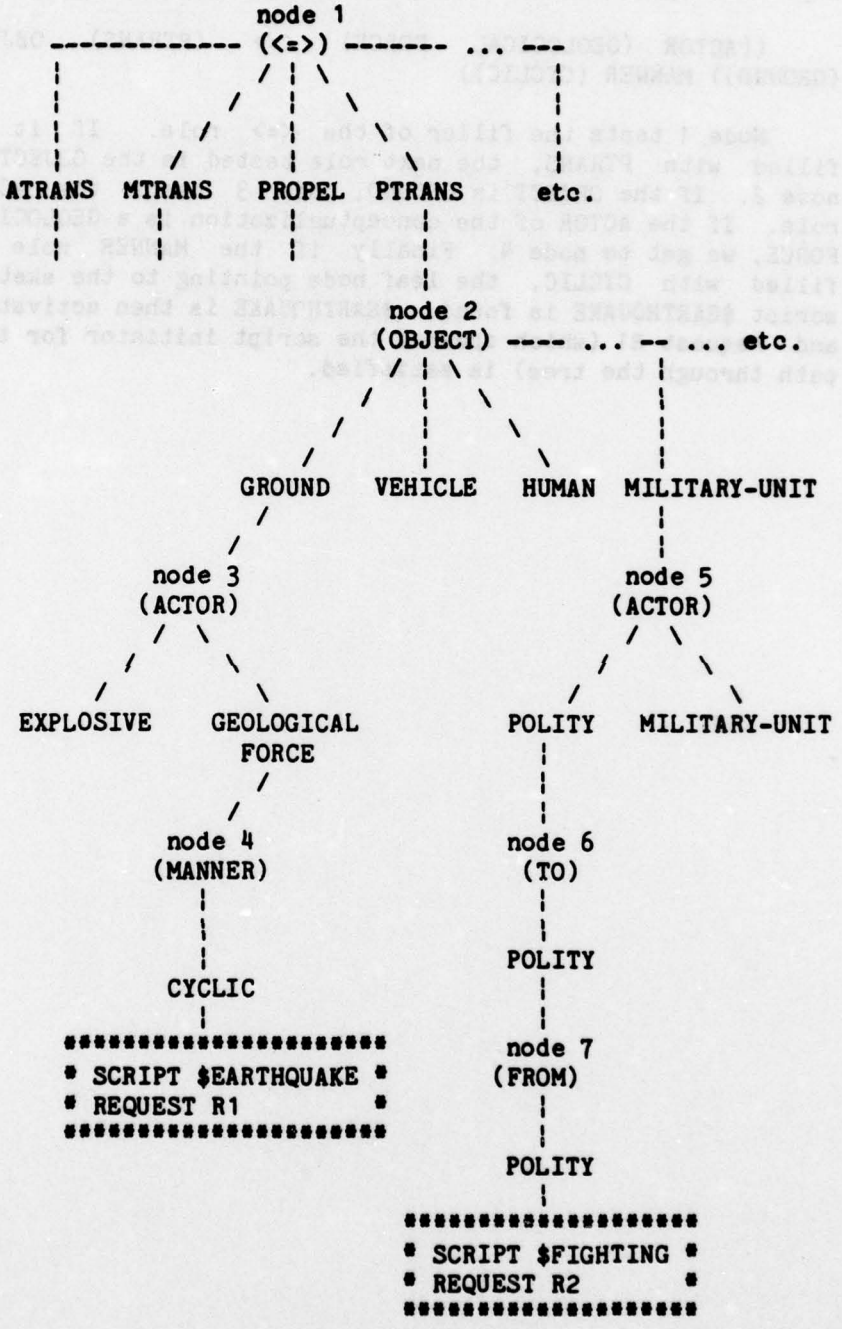


figure 3.1

Figure 3.1 shows the path through the action SSIDT to activate the earthquake and fighting sketchy scripts. The input conceptualization required to activate $EARTHQUAKE is

((ACTOR (GEOLOGICAL FORCE) <=> (PTRANS) OBJECT (GROUND)) MANNER (CYCLIC))

Node 1 tests the filler of the <=> role. If it is filled with PTRANS, the next role tested is the OBJECT at node 2. If the OBJECT is GROUND, node 3 tests the ACTOR role. If the ACTOR of the conceptualization is a GEOLOGICAL FORCE, we get to node 4. Finally if the MANNER role is filled with CYCLIC, the leaf node pointing to the sketchy script $EARTHQUAKE is found. $EARTHQUAKE is then activated, and request R1 (which spawned the script initiator for this path through the tree) is satisfied.

SKETCHY SCRIPT INITIATOR DISCRIMINATION TREE FOR STATES

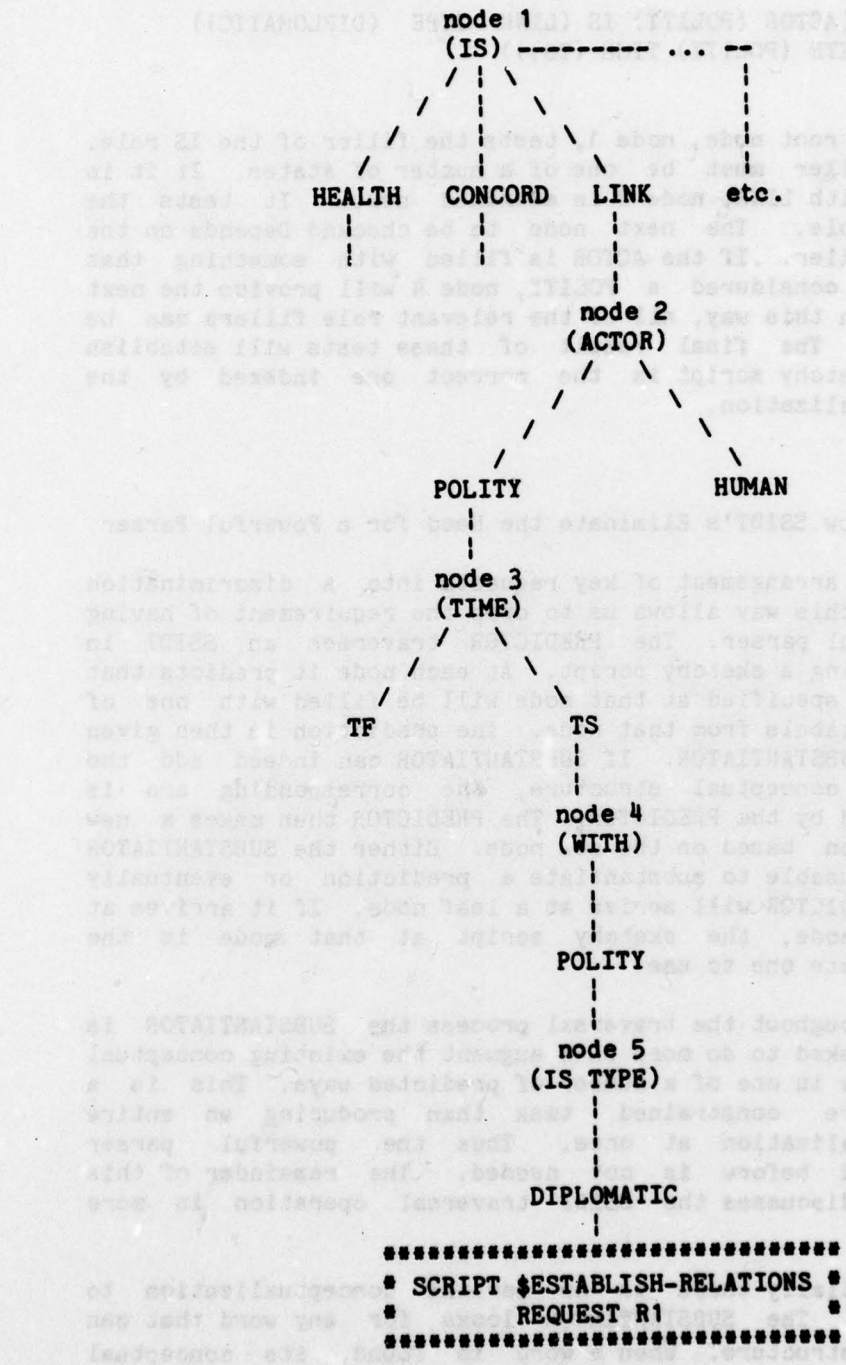


figure 3.2

The above diagram shows the STATE SSIDT. To arrive at the leaf shown the following conceptualization must be built:

((ACTOR (POLITY) IS (LINK TYPE (DIPLOMATIC))
WITH (POLITY) TIME (TS)))

The root node, node 1, tests the filler of the IS role. This filler must be one of a number of states. If it is filled with LINK, node 2 is examined next. It tests the ACTOR role. The next node to be checked Depends on the ACTOR filler. If the ACTOR is filled with something that can be considered a POLITY, node 4 will provide the next test. In this way, all of the relevant role fillers can be tested. The final result of these tests will establish which sketchy script is the correct one indexed by the conceptualization.

3.4.4 How SSIDT's Eliminate the Need for a Powerful Parser

The arrangement of key requests into a discrimination tree in this way allows us to drop the requirement of having a powerful parser. The *PREDICTOR* traverses an SSIDT in identifying a sketchy script. At each node it predicts that the role specified at that node will be filled with one of the arc labels from that node. The prediction is then given to the SUBSTANTIATOR. If SUBSTANTIATOR can indeed add the desired conceptual structure, the corresponding arc is traversed by the PREDICTOR. The PREDICTOR then makes a new prediction based on the new node. Either the SUBSTANTIATOR will be unable to substantiate a prediction or eventually the PREDICTOR will arrive at a leaf node. If it arrives at a leaf node, the sketchy script at that node is the appropriate one to use.

Throughout the traversal process the SUBSTANTIATOR is never asked to do more than augment the existing conceptual structure in one of a number of predicted ways. This is a far more constrained task than producing an entire conceptualization at once. Thus the powerful parser discussed before is not needed. The remainder of this section discusses the SSIDT traversal operation in more detail.

Initially there is no partial conceptualization to augment. The SUBSTANTIATOR looks for any word that can build a structure. When a word is found, its conceptual structure is used as the partial conceptualization. The PREDICTOR examines the new partial conceptualization to determine what kind of conceptual dependency representation

it is. The PREDICTOR selects the appropriate SSIDT for the representation. If the partial conceptualization is an action, the PREDICTOR selects the ACTION SSIDT. If the conceptualization is a state, the PREDICTOR selects the STATE SSIDT, and if it is a state change, the STATE-CHANGE SSIDT.

At any point in the selection process the PREDICTOR examines one node of the SSIDT. This is the current node. Initially the current node is the root node of the SSIDT. The PREDICTOR asks the SUBSTANTIATOR to add the conceptual role specified at the node and predicts that the filler will match one of the arc labels emanating from the node.

For example, suppose the partial conceptualization built were the action (<=> PTRANS). Then the PREDICTOR would select the ACTION SSIDT and node 1 would become the current node. The PREDICTOR would then follow the tree as far as possible. Since the <=> role is already filled with PTRANS, the predictor would make node 2 the current node. It would then predict that the OBJECT role would be filled with one of GROUND, VEHICLE, HUMAN, etc. The SUBSTANTIATOR might fill the OBJECT role with, say, Los Angeles. The PREDICTOR would then interpret Los Angeles as a type of GROUND and follow the arc to node 3 and request that the ACTOR role be filled.

The processing continues until a leaf of the SSIDT is reached. At each leaf is a pointer to the sketchy script that ought to be activated, and the request within the sketchy script that the constructed conceptualization will satisfy. This sketchy script is then activated. The information gleaned from building the key request is incorporated into the sketchy script.

3.4.5 An Example of Event Induced Activation

The following output illustrates FRUMP's processing during an event induced script activation. The input sentence is "Israel has sent troops into Lebanon". FRUMP uses the action SSIDT given in figure 3.1 to activate the sketchy script $FIGHTING. In the example below, the computer output generated by FRUMP is on the left; explanatory comments are on the right. The point to be illustrated here is how the PREDICTOR uses the SSIDTs to guide the SUBSTANTIATOR. The actual workings of both the PREDICTOR and SUBSTANTIATOR are discussed in detail in the next two chapters.

Input:

ISRAEL HAS SENT TROOPS INTO LEBANON.

COMPUTER OUTPUT	COMMENTS
SUBSTANTIATOR: ((<=> (*PTRANS*)) BUILT FROM WORD# (4) WORD SENSE SEND1.	SUBSTANTIATOR has found the word "SENT" which can build a conceptual structure. That structure is submitted to the ACTION Sketchy Script Initiator Discrimination Tree because it is a Conceptual Dependency Action.
PREDICTOR: PREDICTING ROLE (OBJECT) WILL BE FILLED WITH AN ELEMENT FROM THE LIST (*GROUND* *VEHICLE* *HUMAN* *MILITARY-UNIT* *PHYSOBJ*)	After selecting the ACTION SSIDT, the PREDICTOR begins to follow its branches. It follows the PTRANS arc from the node that tests the filler of the <=> role (node 1 in figure 3.1). Now the PREDICTOR is at node 2 which tests the OBJECT role. The PREDICTOR simply predicts the OBJECT will be filled with one of the arc labels leading to a deeper node. If the OBJECT cannot be filled with one of these, the conceptualization is of no interest to the PREDICTOR at this point since it cannot possibly be the key request of a sketchy script.
SUBSTANTIATOR: PREDICTING (OBJECT) IS VERB-OBJECT OF (SEND1 4 NIL PAST)	The SUBSTANTIATOR has decided where in the sentence to look for the filler of the OBJECT role.
FOUND POSSIBLE (*MILITARY-UNIT*) FROM WORD# (5) (OBJECT) HAS BEEN FILLED WITH (*TROOPS*)	It looks there and finds the word "TROOPS" which can indeed be interpreted as one of the predicted fillers - MILITARY-UNIT.
PREDICTOR: PREDICTING ROLE (ACTOR)	The PREDICTOR follows the

WILL BE FILLED WITH AN ELEMENT FROM THE LIST (*POLITY* *MILITARY-UNIT*)	MILITARY-UNIT arc from node 2 to node 5. Node 5 tests the ACTOR role. The only arcs node 5 are labeled with POLITY and MILITARY-UNIT.
SUBSTANTIATOR:	
PREDICTING (ACTOR) IS SUBJECT OF (SEND1 4 NIL PAST)	Again the SUBSTANTIATOR decides where in the sentence to look for the desired conceptual role.
FOUND POSSIBLE (*POLITY*) FROM WORD# (2) (ACTOR) HAS BEEN FILLED WITH (*ISRAEL*)	In the position where it expects to find the syntactic subject of SEND, it finds "ISRAEL" which can be interpreted as a POLITY. Thus the POLITY arc from node 5 is followed.
PREDICTOR:	
PREDICTING ROLE (TO) WILL BE FILLED WITH AN ELEMENT FROM THE LIST (*POLITY*)	The PREDICTOR has now arrived at node 6 of the SSIDT. That node tests the TO role. As the PREDICTOR traces deeper into the SSIDT, there are fewer arcs leading from each node. As FRUMP fills out the conceptualization, there are fewer and fewer key requests that it can partially match. There is only one arc from node 6.
SUBSTANTIATOR:	
WORD# (6) INTO CAN POSSIBLY ADD (TO)	The SUBSTANTIATOR decides that the TO role can be added
TENTATIVELY RESOLVING INTO TO INTO2	by the preposition INTO.
PPEDICTING (TO) IS (PREP-OBJECT) OF (INTO2 6)	It looks where it expects the object of the preposition to be,
FOUND POSSIBLE (*POLITY*) FROM WORD# (8) (TO) HAS BEEN FILLED WITH (*LEBANON*)	and finds the word LEBANON which can be interpreted as a POLITY.
PREDICTOR:	
PREDICTING ROLE (FROM) WILL BE FILLED WITH AN ELEMENT FROM THE LIST (*POLITY*)	By following the POLITY arc from node 6, the PREDICTOR arrives at node 7 of the SSIDT. This node tests the

	FROM role. The only arc leading from the node is labeled with POLITY. Thus the FROM role must be filled with a POLITY if a sketchy script is to be identified.
SUBSTANTIATOR:	
TEXT ANALYZER UNABLE TO ADD ADD (FROM) - CALLING INFERENCE PROCEDURES	The SUBSTANTIATOR is unable to add the FROM role from the text. It therefore resorts to its inferencer.
. . .	
INFERRING (FROM) IS (*ISRAEL* CERTAINTY (8))	The SUBSTANTIATOR is able to infer that the FROM role can be filled with ISRAEL. The actual workings of the inferencer will be discussed in detail in chapter 5. For now it is sufficient to know that the FROM role was filled by the location connected the ACTOR role. This filler is given a certainty of 8 on a scale of 1 to 10 since the inference might be wrong.
PREDICTOR:	
SELECTED SKETCHY SCRIPT $FIGHTING	The POLITY arc is followed from node 8. It leads to a terminal node containing that points to a request within the sketchy script $FIGHTING. The $FIGHTING script is then activated and request R2 is satisfied.

So finally a sketchy script is identified. During the process the SUBSTANTIATOR has always been given top-down predictions to help in analyzing the text in spite of the fact that there was no initial current context. The ACTION Sketchy Script Initiator Discrimination Tree was used by the PREDICTOR to lead the SUBSTANTIATOR through the process of building up a conceptualization capable of activating a sketchy script.

The conceptualization built by FRUMP is

```
(ACTOR      ISRAEL
 <=>        PTRANS
 OBJECT     TROOPS
 TO         LEBANON
```

```
FROM            ISRAEL)
```

Request R2 of $FIGHTING which the above conceptualization matches is

```
(ACTOR          POLITY
 <=>            PTRANS
 OBJECT         MILITARY-UNIT
 TO             POLITY
 FROM           POLITY)
```

In addition the sketchy script $FIGHTING imposes a requirement that the filler of the TO role be different from the filler of the ACTOR role. This is necessary to eliminate the possibility of the script being called up when MILITARY-UNITs are returning to their own country. This test is, of course, satisfied for the conceptualization built in above example.

3.4.6 Complexity of Event Induced Activation

The number of tests needed to activate a sketchy script from a conceptualization is equal to the depth of its leaf. The depth of the leaves is not related to the total number of sketchy scripts in the system but rather the number of conceptual roles in the key request conceptualization; only in rare cases is a role filler examined more than once. The number of sketchy scripts does, however, affect the branching factor of the tree. However, the branching factor is only logarithmic in the number of scripts. Furthermore, most of the work in selecting an arc is done in filling a missing conceptual role not in selecting the arc once the filler has been found. The matching process to select an arc based on how the missing role was filled is very inexpensive. Thus, the complexity of sketchy script activation is logarithmic in the number of scripts in the system and has a very small constant.

This method of script selection is very dependent on having a primitive decomposition of word meanings. It is absolutely essential that the process of matching a conceptualization built from the text to the key requests be spread throughout the discrimination tree. Without that property, the matching process would grow linearly with the number of key requests. This would spell ultimate disaster for the entire system. FRUMP uses Conceptual Dependency decomposition. However, any other decomposition method of similar representational power would also be adequate.

It is possible from the SSIDTs to understand why key word systems work as well as they do. A semantically rich word (whose meaning representation is very nearly an entire conceptualization) can all alone arrive at or very near a leaf node of the SSIDT all by itself. In a key word system, these semantically rich words are explicitly tagged with situational specific information. This circumvents the need for tracing through a structure like the SSIDT. However, in key word systems, problems arise with words that have several meanings and when there is no single semantically rich key word. FRUMP's SSIDTs are general enough that both of these situations are easily handled. Semantically rich words are simply treated as any other word. The difference is that they give a lot of information to the SSIDT so that much progress can be made toward a leaf node.

CHAPTER 4

PREDICTING CONSTRAINTS

4.1 Introduction

Once a script has been selected and a current context has been established, the PREDICTOR can begin making its predictions about what will occur next. This chapter classifies the types of predictions that the PREDICTOR can make and gives the rules for generating them. The next chapter will discuss how the SUBSTANTIATOR services each prediction.

4.2 Kinds of Understander Predictions

There are six broad categories of predictions the PREDICTOR can make. This chapter will discuss how they differ and situations in which each prediction type is made.

The PREDICTOR can predict:

1) a specific sketchy script

2) constraints on a sketchy script

3) one or more particular conceptualizations

4) one or more general constraints on a particular role filler of a conceptualization

5) one precise filler for a role in a conceptualization

6) several precise role fillers for a role in a conceptualization

4.3 Predictions from Issue Skeletons

Often a news article will refer to several related sketchy script situations. Thus while processing a story FRUMP must be able to realize that an additional sketchy script situation is being referred to by the text. When this happens the new sketchy script must be activated to enable FRUMP to process any events from that situation that might be reported.

The PREDICTOR is able to anticipate the related sketchy scripts of a news situation by means of issue skeletons. This is implicit reference activation which was discussed in chapter 3. Here we will outline at a lower level what predictions are actually made during implicit reference activation.

The natural disaster issue skeleton looks like this:

NATURAL DISASTER ISSUE SKELETON

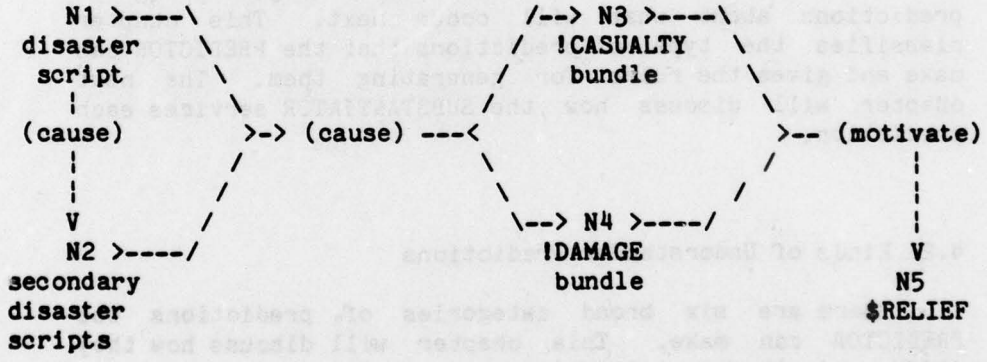


figure 4.1

N1, N2, N3, N4, and N5 are nodes in the issue skeleton. Sketchy scripts can be hooked to the nodes. There are limitations at each node to what can be hooked there.

Node N1 is the node for the initial disaster. Any instantiated disaster script can be hooked at that node.

Node N2 is the node for disasters brought on by the first one. Any number of disaster scripts can be hooked to node N2. However, they must be secondary disasters, that is, disasters which have causes. Natural disasters are not acceptable. The issue skeleton links the initial disaster script to the secondary disaster scripts with a "causes" arrow. This indicates that the secondary disasters result from the initial disaster.

Node N3 must be an instance of the casualty bundle which is a type of request bundle. A request bundle is a collection of conceptual requests similar to a sketchy script. Requests are formed into bundles when that collection of events occur in many different script situations. For efficiency, rather than listing these events once in each script, they are assembled into a unit. Each script that requires these events includes a pointer to the bundle. Thus each event of the bundle is stored only once instead of once for each sketchy script in which it can appear.

There is another reason for using bundles. When several disasters strike the same location news articles do not give separate casualty figures for each disaster. Rather they give one figure for the number dead, one for injured, and one for homeless. There is often no way to apportion these numbers among the disasters. So instead of connecting a casualty figure with each disaster sketchy script, FRUMP must have a way to associate the figures with the news issue as a whole. Therefore, instead of being pointed to from the disaster sketchy scripts, the !CASUALTY bundle is a node in the issue skeleton. It is often the case that bundles are connected to issue skeletons rather than directly to sketchy scripts. This will be discussed further in chapter 8. To differentiate the names of request bundles from sketchy scripts, the names of bundles are preceded by "!" whereas the names of sketchy scripts are preceded by "$". Request bundles behave exactly as sketchy scripts except they do not directly correspond to a specific real world situation.

The casualty bundle contains conceptualizations for the number of people killed, the number of people injured, and the number of people left homeless. Node N3 in the issue skeleton represents the fact that the killed, injured, and homeless are a result of the disaster represented at node N1.

Node N4 has attached to it the bundle !DAMAGE, which is like the casualty bundle but contains events related to property damage. The issue skeleton represents that the bundle hooked to N4 is also caused by the disaster at N1.

The final node N5 can only be hooked to an instance of the sketchy script $RELIEF. This part of the issue skeleton represents that the damage and personal injury may motivate instances of relief aid to the devastated location.

Thus issue skeletons provide a simple method of representing the static relation between sketchy scripts in a news issue.

When an article or set of articles which refer to several related sketchy scripts is read, the internal representation of the sketchy scripts must reflect the relation between them. The conceptual representation of several related sketchy scripts is an instantiated issue skeleton. In an instantiated issue skeleton instances of sketchy scripts and request bundles are hooked to the nodes.

An instantiated natural disaster issue skeleton might look like this:

AN INSTANTIATED NATURAL DISASTER ISSUE SKELETON

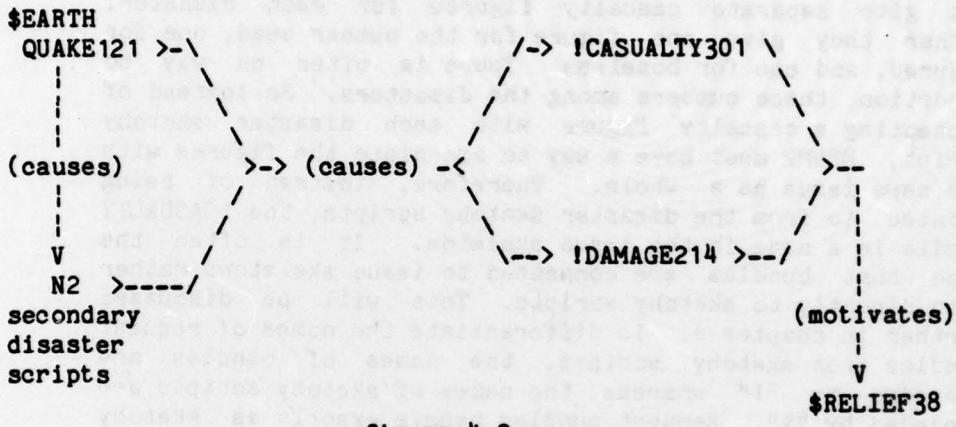


figure 4.2

Here $EARTHQUAKE121 is the particular instantiated script which is the main disaster. For example, $EARTHQUAKE121 might represent the earthquake that struck southern Mexico on August 13, 1975. In this instantiated issue skeleton there are no secondary disasters yet. The instantiated damage bundle !CASUALTY301 has been hooked to node N3. !CASUALTY301 contains the conceptual dependency representations of the number killed, injured, and homeless for $EARTHQUAKE121. If there were an instance of a secondary disaster !CASUALTY301 would reflect those casualty figures as well. !DAMAGE214 has been hooked to N4. It is a particular instance of the !DAMAGE request bundle. $RELIEF38, an instantiated relief sketchy script, has been hooked to N5.

While reading news stories, FRUMP tries to build instantiated issue skeletons. However, before an issue skeleton can be instantiated, it must be <u>initiated</u>. Initiating an issue skeleton means that the system has decided that the input article is discussing the corresponding news issue. From then on, the system tries to hook instantiated sketchy scripts from the news article into

that issue skeleton. This results in an instantiated issue skeleton like the one described above.

The PREDICTOR initiates an issue skeleton when evidence from the current context indicates that the issue skeleton is appropriate. This evidence comes from sketchy scripts activated by FRUMP's bottom-up activation procedures discussed in chapter 3. Some sketchy scripts are marked to indicate that they typically fit into a certain issue skeleton. When one of these sketchy scripts is instantiated, the corresponding issue skeleton must be initiated.

PREDICTOR Rule 1

> When a new sketchy script is instantiated that cannot be used by the current context, and if that sketchy script typically fits into a particular issue skeleton, initiate that issue skeleton.

This rule says that when a sketchy script is instantiated which the PREDICTOR cannot incorporate into an existing issue skeleton, it should check whether this sketchy script can initiate an issue skeleton. If the newly instantiated sketchy script can initiate an issue skeleton, the PREDICTOR creates a token of that issue skeleton and connects the new sketchy script to it.

For example, suppose there are no initiated issue skeletons and FRUMP reads the following story:

> A moderately strong earthquake struck the southern Columbian city of Neiva early today, killing at least 70 and injuring many more. The Columbian seismological station said the quake hit at 3:22 A.M. and measured 7.5 degrees on the Richter scale. The epicenter was 60 miles southwest of Bogota.

FRUMP's activation procedures will identify this story as reporting an earthquake. That is, the sketchy script $EARTHQUAKE will be activated. As soon as any script variable is bound (e.g., when FRUMP identifies, say, the location to be Neiva or southern Columbia) the script is marked as instantiated. The PREDICTOR knows that $EARTHQUAKE is a disaster sketchy script. Therefore, it typically appears in the natural disaster issue skeleton. Since there is no previous issue skeleton that might help explain an earthquake in Columbia, rule 1 applies.

Therefore, the PREDICTOR makes a copy of the natural disaster issue skeleton and connects the newly instantiated earthquake sketchy script with it at node N1.

Issue skeletons provide the PREDICTOR with a means of anticipating sketchy scripts. Notice that in the uninstantiated natural disaster issue skeleton N3, N4, and N5 can only be hooked to specific sketchy scripts or request bundles. N3 can only be matched by an instance of !CASUALTY; N4 can only be matched with an instance of !DAMAGE; N5 can only be matched with an instance of $RELIEF. These specific sketchy scripts and request bundles ought to be expected once the issue skeleton is initiated.

PREDICTOR Rule 2

When an issue skeleton is initiated, predict (activate) the sketchy scripts and request bundles explicitly required by that issue skeleton.

This rule is the underlying mechanism for implicit reference activation described in chapter 3. The rule tells the PREDICTOR to activate a sketchy script whenever a context is built which requires that particular sketchy script.

For example, once the issue skeleton is initiated for the Neiva earthquake, the PREDICTOR activates the sketchy script $RELIEF via rule 2. Thus FRUMP will be able to process the story if it continues:

> The United States announced that the army would air lift foodstuffs and drinking water to the devastated area.

$RELIEF, which is already active, provides the correct context in which to interpret the above sentence. If $RELIEF were not active at this point, it would have to be activated by FRUMP's bottom up mechanisms before the air lift event could be understood. However, rule 2 allows the PREDICTOR to immediately select the appropriate sketchy script. The PREDICTOR can also hook the newly instantiated $RELIEF sketchy script into the correct issue skeleton as well.

The previous discussion described how issue skeleton nodes which required specific sketchy scripts could help the PREDICTOR anticipate input events. However, nodes in an instantiated issue skeleton can be satisfied even after the

article that initiated it has been finished. Later articles can report on further developments of news issues.

After initiating an issue skeleton from a news article, FRUMP must continue to be on the lookout for sketchy scripts to connect to this issue skeleton even after the initiating article has been read. For example, shortly after reading about $EARTHQUAKE121 an article might be read which reports fires from ruptured gas mains in the area of the earthquake, FRUMP must be able to connect the fire to the instantiated issue skeleton of figure 4.2. The instantiated fire script must be hooked into node N2.

PREDICTOR Rule 3

When a sketchy script has been instantiated, check if it satisfies a node in some existing initiated issue skeleton. If so, hook the sketchy script into that issue skeleton.

This rule is equivalent to predicting that the script situations required by an initiated issue skeleton are likely. These predictions enable FRUMP to relate later news articles to a previous news issue. Different nodes in the same issue skeleton can often be satisfied by separate news articles. For example, several days after the report of a flood there may be a report of a cholera epidemic. It is important for the system to realize that the disease is a secondary disaster of the flood.

If FRUMP does not anticipate the possibility of secondary disasters from the flood, it will misinterpret the cholera outbreak as a separate disaster. This will result in two unconnected natural disaster issue skeletons: one for the flood and one for the cholera.

FRUMP must somehow expect the cholera outbreak from the flood article. The PREDICTOR is able to expect these events by using rule 3. Whenever a sketchy script is instantiated, it is tested against the outstanding requirements of the currently initiated issue skeletons. The tests consist of constraints on the type of sketchy script (e.g., for N2 it has to be a secondary disaster; for N5 it must be a $RELIEF sketchy script) and on constraints on the script variables (e.g., for N5 the relief must be sent to the same location as was hit by the disaster).

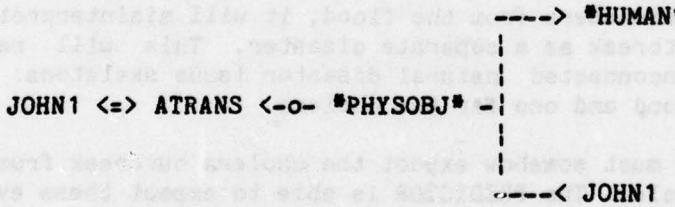
The tests are evaluated whenever a sketchy script that was initiated by either of the bottom up processes (explicit reference or event induced activation) is instantiated. If a script matching all of a test's requirements is found, the

PREDICTOR hooks that script into the appropriate issue skeleton node. If the node requires only one script, then the node is marked as satisfied and that test is removed so no other sketchy scripts will be found. If, however, the node can handle more than one script (as N2 in figure 4.1), the test is continued. Any further sketchy scripts that satisfy the test are also hooked to that node.

Thus rules 1-3 enable the PREDICTOR to anticipate issue skeletons and sketchy scripts associated with news issues. Rule 1 tells the PREDICTOR how and when to initiate an issue skeleton. Rule 2 enables the PREDICTOR to anticipate specific sketchy script situations while processing the story that initiated the news issue. Rule 3 allows the PREDICTOR to correctly relate non-specific sketchy script situations to the proper issue skeleton, and to correctly hook script situations occurring in later news articles to previous issue skeletons.

4.4 Predicting Conceptualizations

When the PREDICTOR decides from the current context that a particular event, state, or state change will occur, it makes a specific prediction of that conceptualization to the SUBSTANTIATOR. A predicted conceptualization is a conceptual dependency structure. In the structure the role fillers might be specific tokens of objects or they might be only type constraints on the ultimate filler of the role. For example, the following CD structure might be predicted:



This prediction tells the SUBSTANTIATOR to expect an event of JOHN1 giving something to someone. JOHN1 is a token of a specific person that FRUMP already knows about.

The SUBSTANTIATOR's job is then to find from the text or infer a specific event which matches this prediction and flesh out the CD structure as much as possible. That is, it will try to find exactly what the physical object is and to

whom it is given.

Conceptualization predictions always come from sketchy scripts or request bundles. Sketchy scripts contain all of FRUMP's knowledge about how the world behaves. It is this knowledge that the PREDICTOR uses to predict likely conceptualizations. There are three ways the PREDICTOR can anticipate a conceptualization. For each way there is a corresponding rule in the PREDICTOR.

PREDICTOR Rule 4
When a sketchy script is activated, predict the default track conceptualizations in that sketchy script.

This is what actually happens when a sketchy script is activated. The rule instructs the PREDICTOR that the normal events of a sketchy script should be predicted when the sketchy script is identified.

For example, consider the following sentence:

The police arrested John Smith early today.

As discussed in chapter 3, the above input activates the sketchy script $ARREST by explicit reference. Among the usual events in $ARREST is the suspect being charged with a crime. Using rule 4 the PREDICTOR predicts to the SUBSTANTIATOR that a likely event is that John Smith will be charged with a crime. These predictions are made in the form of a conceptual dependency representation of an act, state or state change. The SUBSTANTIATOR uses the predictions to guide its text analyzing and inference procedures.

PREDICTOR Rule 5
When a sketchy script variable is bound, check whether the binding can predict non-default conceptualizations.

Some conceptualizations should be predicted only when certain information is found to indicate that they are likely. These are conceptualizations which can occur in the script situation but do not occur often enough to justify always predicting them. For example, one event that can occur during earthquakes is the collapse of buildings. Even though this is a common occurrence most earthquakes reported

are simply not violent enough to destroy buildings. the PREDICTOR therefore only predicts destruction of buildings if the current context indicates a strong earthquake.

To illustrate the rule, consider the following story:

> A 6.3 Richter scale earthquake struck the southern Mexico city of Oaxaca today. Early reports indicate that as many as 100 people have died, many as the result of the collapse of three high-rise apartment complexes. The quake, which is said to be the strongest in 10 years was felt as far north as Mexico City.

When FRUMP recognizes that this is a story about an earthquake, it brings in the sketchy script $EARTHQUAKE. This script contains the fact that the magnitude of the quake will usually be given in the story. From rule 4, the PREDICTOR then predicts that the text will mention the quake magnitude.

Attached to the script variable for the quake magnitude is the fact that if it is over 3 on the Richter scale or 2 on the Mercali scale there is the possibility of buildings collapsing. This is, of course, a rather arbitrary threshold but it is sufficiently low that smaller earthquakes are not able to topple buildings. Thus when the PREDICTOR is told that the quake magnitude is 6.3 it predicts that there might be the destruction of buildings. Of course, this prediction (like all predictions) might be wrong. If, for example, there is no city near the earthquake site, there will probably not be buildings destroyed. The PREDICTOR still makes the prediction. If there are no buildings at hand, that prediction will simply not be verified.

The earthquake sketchy script predicts destroyed buildings if the magnitude of the quake is high enough but does not insist on there being buildings present. That is, the preconditions used to predict destroyed buildings are incomplete. There is a reason for not performing exhaustive tests before making predictions. Remember that the PREDICTOR's job is only to provide guidance to the SUBSTANTIATOR. The PREDICTOR predicts likely events to help channel the SUBSTANTIATOR's processing. The preconditions that trigger the inference are decided upon when the sketchy script is written. They are chosen to provide the most guidance to the SUBSTANTIATOR with the least expense in evaluating them. In the earthquake example, it is easy to test whether the quake magnitude is over a certain threshold. However, it can be difficult to justify that there are buildings around. If the story states only that

the quake occurred in northern Yugoslavia, for example, it would be relatively expensive to decide whether or not there are buildings near by.

Rule 5 represents a kind of inference. Instead of inferring an event, FRUMP uses this rule to infer a prediction from tests on script variables. There is a problem if the event to be predicted is mentioned in the text before the required script variables are satisfied. For example, the above story might well have started

> Three high-rise apartment complexes have collapsed during an earthquake in the southern Mexico city of Oaxaca today killing as many as 100 people. The quake, said to be the strongest in 10 years, measured 6.3 on the open ended Richter scale.

Rule 5 would not be applied until the second sentence. By that time, the event of the buildings collapsing has already been ignored. This rule buys efficiency but with a concomitant danger of not predicting an event soon enough. In cases where it is important not to miss an unusual event in this way, the event must be made part of the sketchy script. Then it will always be predicted, and it follows that a prediction will then exist before the corresponding event is seen in the text.

PREDICTOR Rule 6

> When the presence of a predicted conceptualization is verified, and that conceptualization indicates which of several tracks in the sketchy script should be followed, predict the conceptualizations along the selected track.

Cullingford [1978] discusses how certain parts of scripty situations can progress in any of several different ways. Each of the various possibilities is represented by a different path through the script. These paths are called <u>tracks</u>.

FRUMP makes use of tracks within its sketchy scripts to eliminate irrelevant predictions. Certain conceptualizations are likely only along specific script tracks. This rule states that these conceptualizations should only be predicted when their respective tracks are

applicable.

The system efficiency is degraded by irrelevant predictions. As we shall see in following sections the SUBSTANTIATOR can satisfy a predicted conceptualization a little at a time by a series of partial matches. Although irrelevant predictions from the PREDICTOR can never be satisfied, the system must spend time processing partial matches for them. The time spent deciding that a match is not possible is wasted. Processing efficiency can thus be increased if the PREDICTOR can avoid making irrelevant and unsatisfiable predictions. Tracks within scripts provide an efficient method of eliminating a large number of irrelevant predictions.

To illustrate how tracks can be used to help control predictions consider the following news article:

> Iranian students marched down the streets of Teheran to the royal residence yesterday protesting the continued rule of the Shah.

Even though demonstrators can be injured while demonstrating, FRUMP does not immediately predict that some demonstrators will be injured. This event is not part of the normal demonstration sketchy script. It exists only along the violent demonstration track of the script. Unless the system has some reason to believe that it is following the violent demonstration track of the sketchy script, injuries to demonstrators should not be predicted.

However, if the following sentence is next:

> As the protesters neared the palace the crowd began to hurl stones and fire bombs.

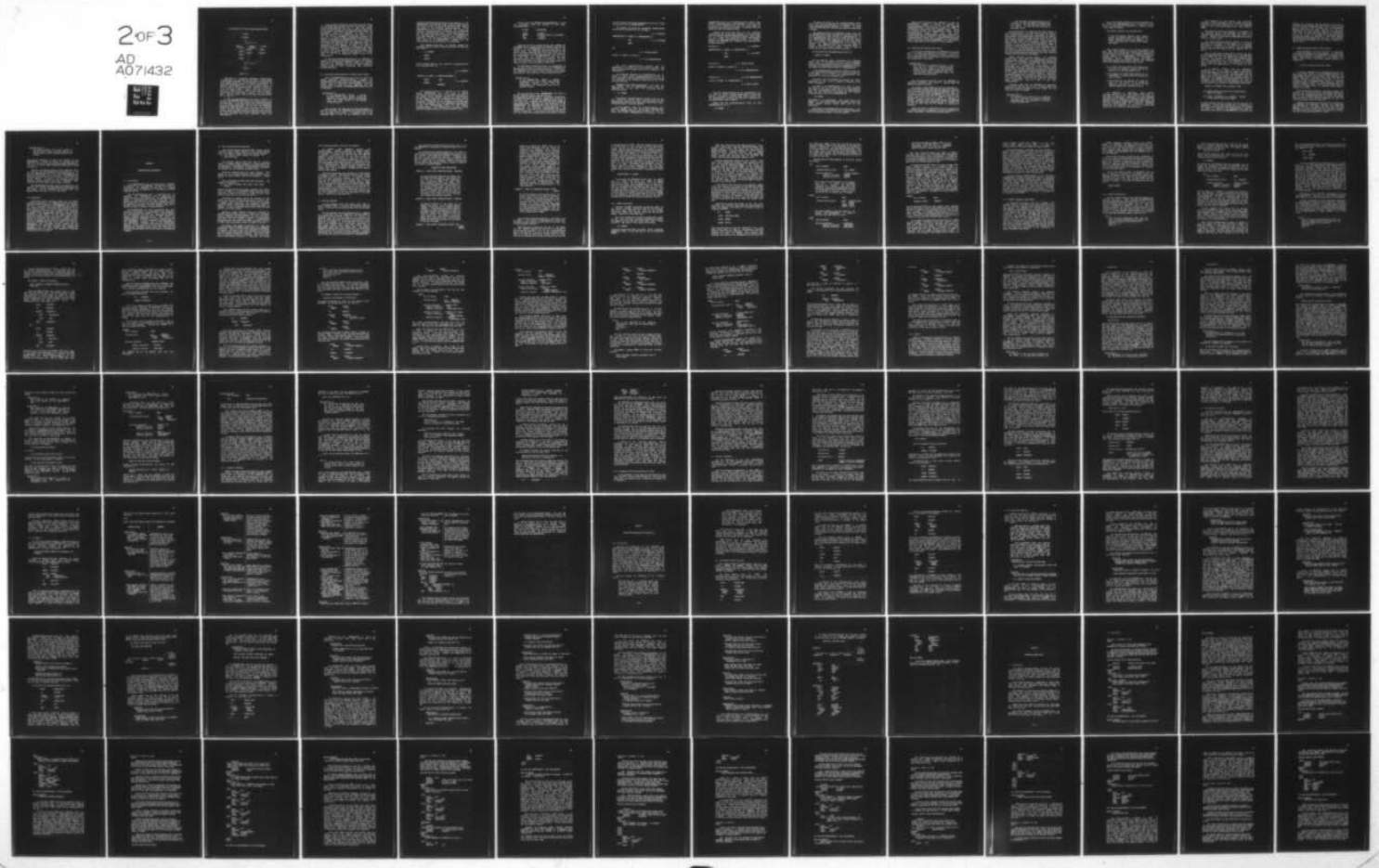
It is clear that the demonstration is following the violent track. Once FRUMP has processed this second sentence it predicts the other events along the violent demonstration track: demonstrators may be arrested, police might fire at them, and there may be injuries and deaths.

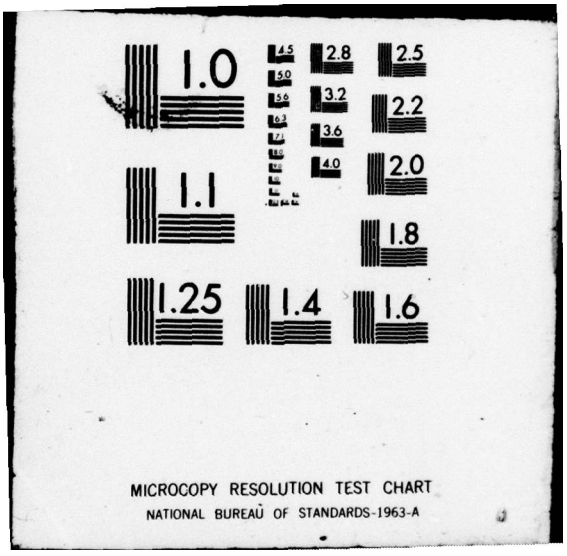
The remainder of this section will be devoted to describing how sketchy scripts are organized into tracks and how the PREDICTOR is able to decide that a particular track is applicable.

The following diagram shows the structure of a multi-track sketchy script:

AD-A071 432 YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE F/G 5/2
SKIMMING STORIES IN REAL TIME: AN EXPERIMENT IN INTEGRATED UNDE--ETC(U)
MAY 79 G F DEJONG N00014-75-C-1111
UNCLASSIFIED RR-158 NL

2 OF 3
AD
A071432




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

The Structure of a Multi-Track Sketchy Script

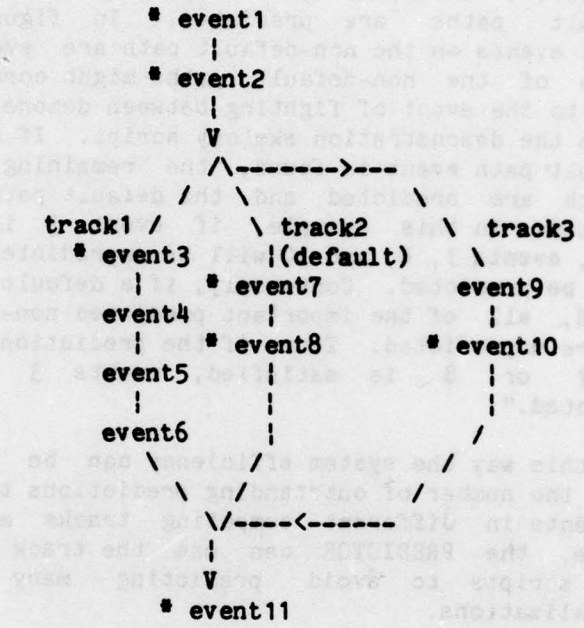


figure 4.3

Each event is a conceptual dependency representation corresponding to an important action, state, or state change in the script situation. In the situation corresponding to this sketchy script, events 1, 2, and 11 usually occur. In addition, either events 3, 4, 5, and 6 or events 7 and 8 or events 9 and 10 will probably occur. However, if events 3, 4, 5, and 6 occur, events 7 and 8 will not occur etc. Events 3-6 make up track 1, events 7 and 8 make up track 2, and events 9 and 10 are track 3. Track 2 is the default path. That is, events 7 and 8 occur more often than events 3-6 or 9 and 10.

In order to take advantage of the track structure in scripts, the PREDICTOR must be able to determine which track is appropriate. It can then predict the conceptualizations for the events along that track and avoid the predictions along other competing tracks. To do this, the PREDICTOR predicts the most important events along each of the various tracks. If one of these is found in the input then the rest of the events along that track are predicted and the events that were predicted from other competing tracks are "unpredicted."

In the script of figure 4.3 the events that are starred are predicted when that sketchy script is activated. These are the events predicted via rule 4. When this sketchy script is activated all of the events along the default path are predicted. In addition, the important events along the non-default paths are predicted. In figure 4.3 the important events on the non-default path are events 3 and 10. One of the non-default paths might correspond, for example, to the event of fighting between demonstrators and police in the demonstration sketchy script. If an important non-default path event is found, the remaining events in that path are predicted and the default path events are unpredicted. In this example, if event 3 is found or inferred, events 7, 8, and 10 will be unpredicted and events 4-6 will be predicted. Conversely, if a default path event is found, all of the important predicted non-default path events are unpredicted. Thus, if the prediction of either event 7 or 8 is satisfied, events 3 and 10 are "unpredicted."

In this way the system efficiency can be improved by keeping the number of outstanding predictions to a minimum. Since events in different competing tracks are mutually exclusive, the PREDICTOR can use the track structure of sketchy scripts to avoid predicting many irrelevant conceptualizations.

4.5 Predicting Characteristics of Possible Role Fillers

Often the SUBSTANTIATOR will not be able to verify an entire predicted conceptualization. Instead, the SUBSTANTIATOR will report back to the PREDICTOR that it has matched a part of a predicted conceptualization. If such a partial match is found then the PREDICTOR has the responsibility of leading the SUBSTANTIATOR through the rest of the match.

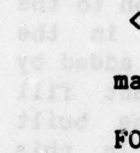
PREDICTOR Rule 7

> If the SUBSTANTIATOR returns a partial conceptualization which matches exactly one prediction, successively predict the remaining roles and their fillers in the matched conceptualization.

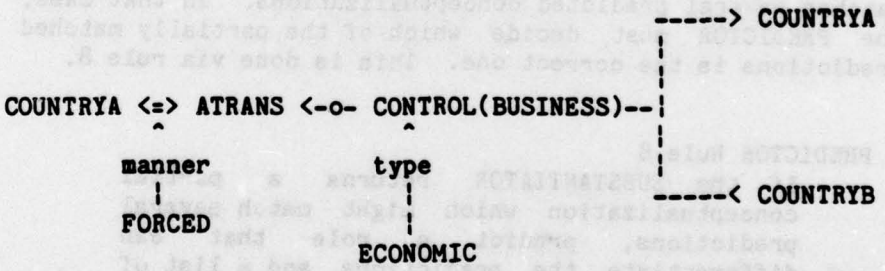
This rule says that when only one conceptualization is partially matched, the PREDICTOR must conclude whether or not an actual match exists. In applying rule 7 the PREDICTOR makes a list of the remaining roles in the

predicted conceptualization and one by one predicts to the SUBSTANTIATOR that each will exist in the conceptualization being built and will be filled with something that matches the filler in the predicted conceptualization. If the SUBSTANTIATOR is able to verify each role and filler, the predicted conceptualization is matched, and the PREDICTOR adds the conceptualization built to the current context. If not, the partial match was a false alarm. The SUBSTANTIATOR never adds something to a structure that contradicts a prediction. The failure of a low level role prediction is never caused by the SUBSTANTIATOR fleshing out the conceptualization the wrong way for the prediction.

As an example of how rule 7 is applied, suppose the following partial conceptualization has been built from the word "took."



Further suppose that the only predicted conceptualization that this might match is:



This conceptualization is predicted in the sketchy script $NATIONALIZE, the script used to process international nationalizations. It represents the action of one country taking economic control of a business from another country. The partial conceptualization was offered by the SUBSTANTIATOR as an attempt at matching this prediction. It matches part of the prediction but the match is far from complete. It is the PREDICTOR's responsibility to help the SUBSTANTIATOR flesh out the partial conceptualization. the PREDICTOR must determine if the predicted conceptualization is actually matched or not.

One at a time, the PREDICTOR asks SUBSTANTIATE to add the following roles and fillers to the built conceptualization:

CD ROLE	FILLED WITH
ACTOR	a country
OBJECT	economic control of a business
TO	a country
FROM	a country

Since the act role is already filled with ATRANS and the MANNER role is already filled with FORCED, these are not predicted. If the SUBSTANTIATOR is able to add each of the desired roles, the PREDICTOR adds that prediction to the current context, indicating that it has been found in the text. If, however, any predicted role could not be added by the SUBSTANTIATOR, or if the SUBSTANTIATOR could not fill the role with the desired role filler, the built conceptualization does not match the prediction. In this case the prediction is not added to the current context but remains so that it may be matched by later text.

The PREDICTOR's responsibility can be complicated if the partial conceptual structure built by the SUBSTANTIATOR matches several predicted conceptualizations. In that case, the PREDICTOR must decide which of the partially matched predictions is the correct one. This is done via rule 8.

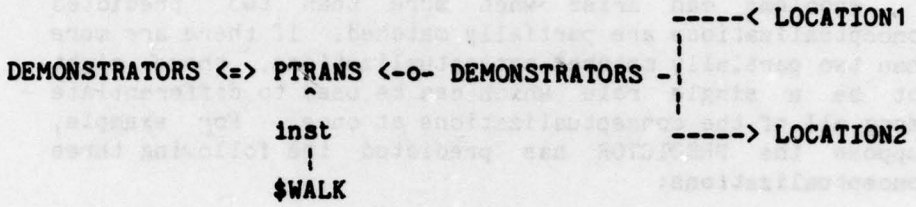
PREDICTOR Rule 8

> If the SUBSTANTIATOR returns a partial conceptualization which might match several predictions, predict a role that can differentiate the predictions and a list of possible fillers.

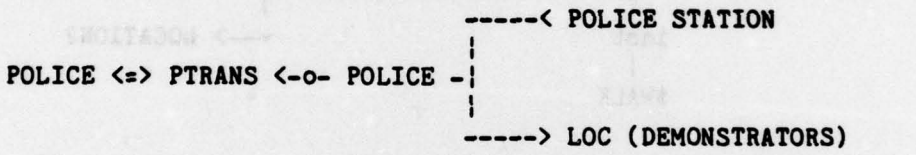
The rule says that if the SUBSTANTIATOR has built a partial conceptualization, and that partial conceptualization matches several outstanding predictions, the PREDICTOR should not predict a single role and filler as dictated by rule 7. Instead, the PREDICTOR should choose a role which has a different filler in each of the predicted conceptualizations. This role is then predicted to the SUBSTANTIATOR. The SUBSTANTIATOR is also given the list of fillers from the predicted conceptualizations. The SUBSTANTIATOR then tries to fill the predicted role with an element from the list. If the role can be added and filled, the only predicted conceptualization that might be matched

is the one whose role filler corresponds to the role filler actually built by the SUBSTANTIATOR.

For example, the script for processing demonstrations makes among its predictions the following:



and



The first conceptualization predicts that the demonstrators will march from one location to another. The second says that police might arrive at the location of the demonstrators.

The conceptual inferencer of the SUBSTANTIATOR may be unable to verify either of these predictions in toto. If this is the case, then the SUBSTANTIATOR must rely on the CD role inferencer or text analyzer to match part of one of the conceptualizations.

Suppose the text analyzer has found a word that can mean PTRANS. Then the SUBSTANTIATOR will inform the PREDICTOR that it has built the following partial conceptualization:

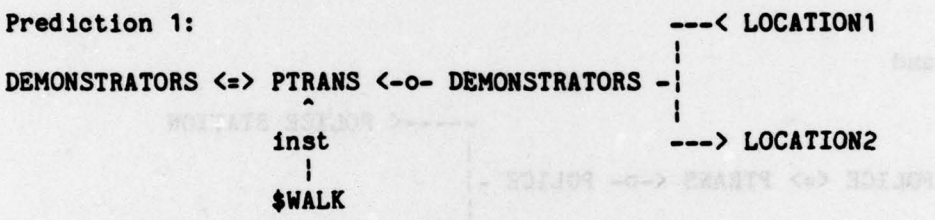
<=> PTRANS

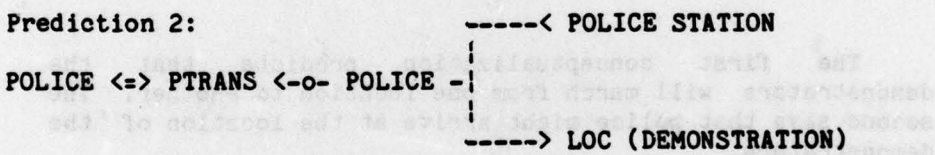
This partial conceptualizations matches both of the above predictions equally well. Rule 8 dictates that the PREDICTOR must predict a role which can differentiate which conceptualization is actually matched.

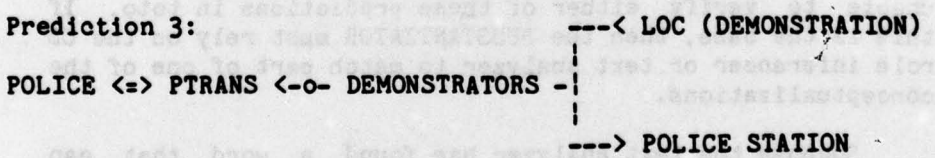
In this example, since the conceptualizations have different ACTORS, that role can be used to differentiate them. The PREDICTOR predicts that the ACTOR role will be filled with either "demonstrators" or "police." If the

SUBSTANTIATOR cannot fill the ACTOR role with either, then neither conceptualization can be matched. If the role can be filled with "demonstrators" then the first might be matched but certainly not the second. If "police" fills the ACTOR role then the reverse is true.

Problems can arise when more than two predicted conceptualizations are partially matched. If there are more than two partially matched conceptualizations, there might not be a single role which can be used to differentiate among all of the conceptualizations at once. For example, suppose the PREDICTOR has predicted the following three conceptualizations:







The third conceptualization is the representation for the event of police bringing demonstrators to the police station. This might be predicted when FRUMP decides that the demonstration is violent and there is a probability of the demonstrators being arrested.

Recall that the conceptualization built by the SUBSTANTIATOR so far is:

<=> PTRANS

No single role added to the conceptualization being built will allow the PREDICTOR to distinguish which of the three predicted conceptualizations is matched. In these cases the PREDICTOR must differentiate among the predictions in stages.

For these predictions, the PREDICTOR might first ask the SUBSTANTIATOR to find the ACTOR role and predict that it will be filled with either "police" or "demonstrators." If the SUBSTANTIATOR is able to add the ACTOR role and fill it with "demonstrators" then prediction (1) is the only viable one. If, however, the SUBSTANTIATOR fills ACTOR with "police" then either predictions (2) or (3) might be matched. The PREDICTOR must then distinguish which, if either, of these is actually matched. This is done by predicting yet another role and list of fillers.

At this point the conceptualization built is:

POLICE <=> PTRANS

The two remaining viable predictions have different OBJECT roles. If the OBJECT role were filled the PREDICTOR could decide which conceptualization was actually matched. Therefore, the PREDICTOR then asks the SUBSTANTIATOR to fill the OBJECT role with either "police" or "demonstrators."

If the SUBSTANTIATOR can build the OBJECT role and fill it with "police" then prediction (2) is the only viable one. If the OBJECT role is filled with "demonstrators" then prediction (3) is the only possible match.

Of course, once the PREDICTOR has narrowed the viable predictions to one, rule 7 is applied to determine if the match is complete.

As always, if the SUBSTANTIATOR is not able to build the role at all, or if it can fill it only with something that cannot be considered either "police" or "demonstrators" none of the predictions can match and the original conceptualization built by the SUBSTANTIATOR

<=> PTRANS

has proved to be inapplicable. This might occur, for example, if the article states that newsmen arrived on the scene. Newsmen arriving cannot match any of the predictions.

The process the SUBSTANTIATOR goes through when a list of possible fillers is predicted for a conceptual dependency role is discussed in detail in the next chapter.

Rules 7 and 8 are used by the PREDICTOR to lead the SUBSTANTIATOR through a process of matching the conceptualization being built to one or more predicted conceptualizations. This matching process is aimed at transforming a partial match to a complete match. During the matching process the SUBSTANTIATOR fleshes out the conceptualization being built until the built conceptualization either completely matches a prediction or clearly cannot match a prediction. The fleshing out of the conceptualization being built is guided by the PREDICTOR. Rules 7 and 8 dictate how the PREDICTOR is to use sketchy script constraints on role fillers of script requests.

4.6 Predicting One Explicit Role Filler

In the proper circumstances the PREDICTOR can do better than to predict constraints on a role filler. If the role filler to be predicted corresponds to a script variable that has already been bound, the binding of that script variable can be predicted in its place.

PREDICTOR Rule 9

> If a role filler is predicted by rule 7, and that role is filled by a previously bound script variable, predict the more explicit binding of the script variable instead of the less specific filler constraint from the predicted conceptualization.

This rule basically says that when the PREDICTOR is predicting individual role fillers it should make the tightest prediction possible.

For example, suppose FRUMP has read the beginning of a story about Iran nationalizing American owned oil interests. From the nationalize sketchy script the PREDICTOR makes the prediction that there might be compensation made. Thus from the generic sketchy script $NATIONALIZE, the PREDICTOR can predict that there will be an ATRANS of money or other compensation from one country to another or companies of another. However, from the current context the PREDICTOR can make the more explicit prediction that the ATRANS will in fact be FROM Iran and TO America or American oil companies.

These more explicit predictions can be of enormous help to the text analyzer, for example, in resolving anaphoric or implied references. Consider the following story:

> Iran announced the nationalization of the holdings of Shell and Mobil oil companies in that country. For the property they were paid $2 million in cash and $12 million in government bonds.

The problem here is to figure out who "they" refers to in the second sentence. The word "paid" means that money or other valuables were given by one participant to another. Understanding this passage entails identifying who paid and who was paid. One way to resolve these roles is to have some memory/inference process notice that the conceptual event of being paid for a possession implies that the receiver of the money gave up the possession. Memory of previously understood inputs could then be searched for an instance of the candidates (Iran or Shell and Mobil) giving up "property." The nationalization event would be found. From this a system could conclude that since Shell and Mobil were the ones who gave up a possession in the past, they are probably the ones receiving the payment.

FRUMP's approach is much simpler. the PREDICTOR predicts that there might be a compensation event in any nationalization situation. The predicted event contains the direction of the ATRANS: the recipient is the participant whose property was nationalized; the donor is the participant who performed the nationalization. These participants are script variables in the nationalization sketchy script and were bound during processing of the first sentence. The compensation event predicted by the PREDICTOR is an ATRANS from Iran to Shell and Mobil. Thus when a compensation input event is found to match the prediction from the PREDICTOR, the preferred participants of the event are already included. Since the text does not contradict these predictions, they are used in the final representation.

There are times when the PREDICTOR can anticipate almost exactly what will fill a role. In these cases if the prediction is explicit enough and strong enough the role filler can simply be assumed.

PREDICTOR Rule 10

> If an explicit role filler is to be predicted by rule 9, and there is previous information indicating that the prediction is correct, assume the prediction.

This rule provides a method by which the PREDICTOR need not bother the SUBSTANTIATOR with a request for a role filler. It says that if the prediction is certain enough, it can simply be assumed.

For example, consider the following story:

> 1) There was renewed fighting today between Israeli and Syrian forces. 2) Syria fired on Israeli positions in the Golan Heights. 3) Israel retaliated with strikes against Syrian fuel dumps.

After reading sentences 1 & 2 a top down processor can make some very certain and precise predictions while interpreting the third sentence. For reasons explained in the next chapter the first word that is examined by the text analyzer section of SUBSTANTIATE is "retaliated." "Retaliated" builds a conceptualization that means an unspecified negative act occurred. However, the dictionary entry also furnishes three other pieces of data.

a) It predicts that the ACTOR of the negative act will be its syntactic subject and was the recipient of a previous negative act.

b) The negative act itself might be found as an instrumental - probably after the preposition "by" or "with."

c) The recipient of the negative act might be found as the object of the preposition "against," and will be the actor of the previous negative act referred to in (a).

The PREDICTOR is immediately given partial conceptualizations as the text analyzer builds them. In this case, it is given the information in the dictionary definition of "retaliate." The system has already processed sentences 1 and 2. Therefore it already knows about the negative act of Syria against Israel. From this information and the information from the "retaliate" dictionary definition, the PREDICTOR can use rule 9 to predict that Israel will be the ACTOR of the negative act.

This prediction is very certain. The dictionary definition stated absolutely that the ACTOR would be the recipient of a previous negative act, and the PREDICTOR knows of only one previous negative act: Syria firing on Israel.

The prediction is also quite precise. As discussed in the previous section the PREDICTOR can predict the constraint that whatever fills the ACTOR role must be a country. However, in this case the PREDICTOR has predicted the actual country itself.

In a similar manner an explicit country can also be predicted for the recipient of the negative act. The system knows that the recipient is the ACTOR of the previous negative act. In this case it is Syria. Thus the PREDICTOR can with some certainty predict that the recipient will be Syria.

Rule 10 states that in cases such as these, the role filling mechanisms need not be called at all. The PREDICTOR all by itself can fill the ACTOR and recipient roles correctly. The benefit is that the rest of the sentence need not be processed at all. If the PREDICTOR is content with interpreting this input as a negative act by Israel against Syria, the system can go on to process the next phrase.

It is possible, however, that the PREDICTOR's assumptions will not be explicit enough to satisfy the script requirements. The script, for example, might dictate that it is important to find not only exactly what the negative act is, but also exactly what military units of Israel are the aggressors and what possession of Syria is the target. In this case, the text analyzer will have to interpret parts of the rest of the sentence, but it can do so more easily in the context of these precise predictions.

Suppose, for example, that sentence 3 read

3a) Israeli planes retaliated with strikes against Syrian fuel dumps.

3b) Israel retaliated by launching aircraft strikes against Syrian fuel dumps.

In these sentences specific military units are given as the aggressors. If the script dictates that it is important to know that the military units are in fact planes as opposed to tanks or missiles, the PREDICTOR must make the appropriate prediction to the SUBSTANTIATOR so that the text

analyzer can be called. The text analyzer is told that it is trying to find the aggressor and that it will be some kind of military units of Israel. From its knowledge of English syntax and the data supplied by the word "retaliate," the text analyzer has only a few places to look. First, it might look at the subject of "retaliated" as in 3a. If it is not there, it is possible that the military units might be specified in the more explicit specification of the negative act as in 3b. In either case, the role filling mechanisms treat the request according to rules 7 and 9.

4.7 Predicting Several Explicit Role Fillers

Often information of the sort supplied by "retaliate" in the previous example will not be present. In these cases it is often impossible for the PREDICTOR to determine alone which script variable corresponds to the desired role filler. Suppose that in the previous example sentence 3 is replaced by

3c) Israel attacked Syrian fuel dumps.

It is still important for the PREDICTOR to furnish the best predictions possible for the SUBSTANTIATOR to use in interpreting the input. In processing this sentence, the system again will initially try to find a word that can build a structure. The first such word in 3c is "attacked." The conceptualization built by "attacked" partially matches a predicted conceptualization from the military fighting script. The script conceptualization is looking for military units of countries doing negative things to each other.

Before processing sentence 3c, the system already has selected the script for military fighting and bound script variables for the fighting countries to Israel and Syria. That is, the PREDICTOR knows from the first two sentences that the story is about military fighting and that the countries involved are Israel and Syria.

Therefore the system can predict explicit countries for the aggressor and recipient. However, the predictions cannot be certain. The system can know only that the aggressor is either Israel or Syria or an ally of either; the same is true for the recipient. To find which is which, the role filling mechanisms must be called.

PREDICTOR Rule 11

> If a single explicit role filler cannot be decided upon, predict that the role will be filled from the list of possible explicit role fillers.

This type of prediction is still very valuable to the SUBSTANTIATOR. Instead of having to construct the role filler for itself, it only has to find enough information in the text to decide which among the predictions is being referred to.

Suppose in processing sentence 3c the PREDICTOR wants to find out who the recipient of the hostile act is. It asks the SUBSTANTIATOR to fill the role corresponding to the recipient with Israel, Syria, or an ally. The PREDICTOR can make the prediction that it will be one of these from the current context. It knows it is reading a story about fighting involving Israel and Syria. Therefore, other hostile acts will probably involve these countries.

The SUBSTANTIATOR determines that the recipient of the negative act is Syria and the possessions being harmed are fuel dumps. Now the PREDICTOR can predict that the aggressor must be Israel or one of its allies.

4.8 Conclusion

Using these eleven heuristic rules and the script activating procedures, the PREDICTOR is able to provide a detailed context in which the SUBSTANTIATOR can interpret the text. The predictions of the PREDICTOR can be made at many different levels. If a higher level prediction of an issue skeleton, sketchy script, or conceptualization cannot be substantiated by the text analysis and inference procedures in the SUBSTANTIATOR, they can be refined by the PREDICTOR to lower levels. At the lowest level, small pieces of conceptualizations are predicted. These low level predictions eliminate the need for a powerful parser. Instead of having to build an entire conceptualization at a time, the SUBSTANTIATOR need only verify role fillers. Furthermore, the refinement process insures that the SUBSTANTIATOR will always have the most explicit predictions available. In the next chapter the methods the SUBSTANTIATOR uses to satisfy the PREDICTOR's predicted constraints will be outlined.

CHAPTER 5

SUBSTANTIATING CONSTRAINTS

5.1 Introduction

It is SUBSTANTIATOR's job to verify and give substance to predictions of the PREDICTOR. There are two levels of predictions. The PREDICTOR can ask that an existing conceptualization be augmented, or it can predict an entire conceptualization.

When the PREDICTOR asks that a conceptualization be augmented, the SUBSTANTIATOR tries to flesh out the conceptualization in the desired way. The SUBSTANTIATOR can augment a conceptualization in either of two ways: it can examine the input text or it can infer conceptual roles. The prediction of an entire conceptualization can only be satisfied by an inference. The text analyzer must use a stepwise process to satisfy a conceptualization.

The SUBSTANTIATOR has three sub-modules that build conceptual structures: a text analyzer, a role inferencer, and a conceptualization inferencer. Two of the sub-modules, the text analyzer and the role inferencer, are used to augment existing conceptualizations. The third, the conceptualization inferencer, is used to build an entire conceptualization. A fourth sub-module interfaces these to the PREDICTOR's requests. This is the SUBSTANTIATOR selection routine. It decides which module to use for which requests. In this chapter I will first describe how each of the structure building modules works and then how the selection routine chooses the correct module to substantiate a request.

5.2 The Conceptualization Inferencer

The conceptualization inferencer makes script related inferences about events implied by but missing from the text. For example, consider the following input sentence:

(1) The United States opened an embassy in Swaziland today.

This sentence implies much more than it literally states. An embassy cannot be opened in a country unless the two countries involved have previously recognized each other diplomatically. Furthermore, it implies that a diplomatic link between the two countries currently exists.

These are inferences from the input sentence. Even though we can be very confident of these inferences if we believe the input sentence, they are not logically entailed by it.

Other situations will have other such inferences. For example, the sentence

(2) The Police charged John Smith with armed robbery.

implies that the police have already arrested John Smith. Thus, if FRUMP is to understand stories about real world situations, it must also be able to infer missing events based on conventions of how the world works.

Each event in a sketchy script has pointers to the events that can be inferred when that event is found. When a conceptual structure is built from the text which matches a sketchy script prediction, the conceptual inferencer checks the script for any other script events that might be assumed. If so, it satisfies those script events and adds them to the current context.

Typically, these inferred events are connected by causation and entailment relations. FRUMP's world knowledge about a situation, its sketchy script for that situation, includes causation and entailment relations to other events that make up the sketchy script. SUBSTANTIATOR's conceptualization inferencer uses these entailment relations to infer other events when an event is found in the text.

For example, with respect to sentence (1), there are two relevant events in FRUMP's sketchy script for diplomatic recognition. There is an event that represents establishing an embassy, and another event that represents two countries being in a state of diplomatic relation. The first event has an inference pointer to the second. Thus if the first event (establishing an embassy) is satisfied, the second

event (having diplomatic ties) will be inferred.

In general these inference pointers are uni-directional. Opening an embassy, for example, implies formal recognition, but recognition does not imply opening an embassy. A country may possibly have just announced recognition of another and not yet had time to open an embassy. Another possibility is that one or both of the countries are too poor to maintain embassies in the other. There are many countries that the U.S. recognizes that do not have embassies here simply because it would be too much of a financial burden. Sentence (2) also illustrates this point. Even though in our judicial system a suspect is not charged until he is arrested, he might well be arrested without being charged.

This type of inference was a mainstay of the SAM system (Cullingford [1978]). SAM worked by finding a path through the possible tracks of its scripts. It inferred events along the path necessary to connect two known inputs. In this way SAM demonstrated that scripts were a useful construct in constraining the inference process. These script inferences are less important for FRUMP because FRUMP is not attempt to demonstrate the utility of scripts but rather an attempt to integrate parsing with the rest of the understanding process. Script inferences have little to do with the parsing process directly. The text analyzer and the role inferencer are of more importance to the FRUMP system. These will be discussed next.

5.3 The Text Analyzer

The text analyzer is the only module that looks at actual English input. It looks at and interprets only one word at a time with three exceptions which will be discussed later.

The text analyzer does its processing only in response to a prediction from the PREDICTOR. Even when FRUMP is selecting an initial sketchy script for an article, the PREDICTOR must help the text analyzer with predictions by means of the Sketchy Script Initiator Discrimination Trees described in chapter 3. As was mentioned in the previous chapter, the PREDICTOR usually anticipates constraints on what might happen next rather than making explicit predictions about what must happen next. In general PREDICTOR's constraints become more specific as FRUMP builds up more context for the article being processed. Specific predictions are more useful than non-specific ones because they provide more direction and guidance to SUBSTANTIATOR.

Only predictions arising from rules 7,8,9, and 11 of the PREDICTOR are candidates for being satisfied by the text analyzer.

The following list contains examples of the kinds of predictions that the selection mechanism might direct to the text analyzer. Each prediction contains a conceptual role and one or more fillers for that role. A prediction may specify a single filler or a list of possible fillers, any one of which will be acceptable. Each of the predicted fillers may be either general (a type of object) or specific (the token of a certain object).

EXAMPLES OF ROLE PREDICTIONS

EXAMPLE 1: ROLE: ACTOR, PREDICTED FILLER: (COUNTRY)

The first prediction asks that the ACTOR role be filled with some country. Any country will do. This is a single general prediction. This prediction might be made while understanding an article about foreign aid. One of the important facts in such stories is the identity of the country giving the aid. In conceptual dependency this would be the ACTOR of an ATRANS. The constraint that it must be a country comes from the international aid script; this script is only concerned with interactions between countries.

EXAMPLE 2: ROLE: ACTOR, PREDICTED FILLER: (CANADA)

The second is similar to the first but it expects the filler to be a specific country (Canada). Prediction (2) is a single specific prediction. This prediction would by made in the same situations as Prediction (1) when the PREDICTOR already has a hypothesis about the country's identity. For example, an article might begin with "The Canadian legislature passed a new foreign aid bill." From this the system can expect that the ACTOR of any ATRANS of aid will be Canada.

EXAMPLE 3: ROLE: OBJECT, PREDICTED FILLERS: (VIP
TROOPS
WEAPON)

Prediction 3 expects the OBJECT role to be filled with something that can either be considered as an important person, military troops, or a kind of weapon. It is a multiple prediction. This prediction could arise in an article about fighting between two countries. Suppose a PTRANS action (an action changing the location of something) has been constructed from the text. The PREDICTOR is interested in three types of PTRANS actions in fighting: An important person going somewhere for peace talks, troops invading or withdrawing, and the PTRANS of weapons as in bombings and shellings. The PREDICTOR would make this prediction to differentiate which of these important events the PTRANS found might be. Of course, this test alone is not sufficient to decide which of the important events is matched. Further predictions and substantiations would have to be made to insure that the conceptualization found completely matches an expected event. This prediction starts the matching process in the right direction.

EXAMPLE 4: ROLE: TO, PREDICTED FILLERS: (SYRIA EGYPT)

The fourth prediction wants the TO role to be filled with either of two specific countries: Syria or Egypt. This prediction might occur in a story about fighting between Israel and the Arab countries of Syria and Egypt. If an instance of Israel launching long range missiles has been constructed, the PREDICTOR is able to utilize the context from what has been understood thus far to predict that the target (the TO role of the PTRANS of missiles) will be either Syria or Egypt.

Each of these predictions asks that a particular role be added to the current conceptualization being built. Furthermore, each requires that the filler of the role have a certain semantic property.

Upon being given a prediction like one of the above, the text analyzer examines the input text in an attempt to add the desired conceptual role to the conceptualization being built and fill it with the desired conceptual object. That is, it looks for some word or phrase from the input that can be interpreted as filling the desired role with one

of the desired role fillers. The text analyzer first tries to decide where in the sentence to look for the desired filler. That is, it tries to find the location in the surface sentence that corresponds to the desired conceptual role. Sentence locations are the syntactic components of a sentence like the subject of a particular verb in the sentence, the object of a particular preposition, etc. Once the text analyzer decides on a particular sentence location, it looks in the neighborhood of that syntactic location for something that can be interpreted as one of the desired items. If a word is found that can be interpreted in that way, it is used. For example, suppose FRUMP is reading a story it knows to be about fighting and sees the input "Israel sent the third army to Damascus." After processing "Israel sent," suppose the conceptual representation built is

(ACTOR ISRAEL <=> PTRANS)

At this point the PREDICTOR will ask that the OBJECT role be filled with either VIP, TROOPS, or WEAPON (Prediction 3 above). The text analyzer will then decide from the verb "sent" that the OBJECT filler can probably be found in the location of the syntactic object of the verb. The text analyzer will then look where it expects to find the syntactic object of the verb and try to interpret what it finds there as an instance of a VIP, TROOPS, or a WEAPON. "The third army" is readily interpreted as a kind of TROOPS so the prediction is satisfied. The rest of this section discusses how the FRUMP text analyzer goes about its job.

5.3.1 FRUMP's Dictionary

Each word in FRUMP's dictionary can have any number of word senses, although typically a word will not have more than two or three. There are two types of word senses: 1)Structure Adding Word Senses, and 2) Role Filling Word Senses. These will be described next.

Part of the definition of Structure Adding Word Senses is that they can build a conceptual dependency structure. For example, the word "go" has a sense which means "change location." This sense of "go" builds the structure

(<=> PTRANS)

A Structure Adding Word Sense can supply filled conceptual dependency roles. In this case the <=> role is filled with PTRANS.

Words that contribute to the underlying action, state, or state change have Structure Adding Word Senses. For example, "shout," "hit," and "throw" all have Structure Adding Word Senses. Their word senses build respectively an MTRANS structure, a PROPEL structure, and a PTRANS structure. Often verbs will have Structure Adding Word Senses. However some non-verbs such as "eruption" and "storm" have Structure Adding Word Senses as well.

Structure Adding Word Senses can also indicate where other role fillers will be found. Again consider the word "go." The Structure Adding Word Sense of "go" builds the <=> role which it fills with PTRANS. It also contains the information that the conceptual ACTOR and the conceptual OBJECT are the same. That is the thing causing the change in location (the ACTOR) and the thing undergoing the change in location (the OBJECT) are the same. Furthermore, the filler of these roles must be animate. The sentence "The table went." is not meaningful with this verb sense of "go." but "John went." is. Finally, a Structure Adding Word Sense can indicate where the new role fillers might be found in relation to the current word.

In the case of "go" the filler of the ACTOR and OBJECT roles can be found in the location of the syntactic subject of the verb. Multiple conceptual roles can be filled from the same syntactic location in the sentence, as in this case where both the ACTOR and OBJECT are filled with the subject. Furthermore, a word sense can add several new roles from different locations in the sentence. If the word sense had information on where, say, the TO role could be found, that could be incorporated just as the ACTOR and OBJECT roles were.

Some Structure Adding Word Senses do not both build structures and indicate where other role fillers might be found. For example, a word sense of "earthquake" builds the following structure:

```
(<=>     PTRANS

 ACTOR   GEOLOGICAL-FORCE

 OBJECT  GROUND

 MANNER  CYCLIC)
```

This word sense does not have any information about where other roles might be found in the sentence. Verbs can indicate where certain conceptual role fillers will typically be found. For example, the word sense of "go" discussed above contains the information that the filler for

the ACTOR and OBJECT roles will be found as the verb's syntactic subject. Nouns, like earthquake, do not make such syntactic predictions. On the other hand the word "last," in its verb sense as in "the storm lasted three days," does not itself build a structure but can indicate where a new role filler may be found; the conceptual role DURATION can be filled with what is found in the syntactic position of the object of the verb.

The following are three examples of Structure Adding Word Senses.

GO1:

Part Of Speech:	VERB
Conceptualization Built:	(<=> PTRANS)
New Role Locations:	((ACTOR) (OBJECT))
Semantic Constraint:	ANIMATE
Syntactic Location:	SUBJECT

GO1 is one of the word senses of the English verb "go." It builds the conceptual structure (<=> PTRANS). The New Role Locations of this word sense indicate that the conceptual roles ACTOR and OBJECT may be found as the SUBJECT of this verb. There is also a constraint that the SUBJECT must be animate.

QUAKE1:

Part Of Speech:	NOUN
Conceptualization Built:	(ACTOR GEOLOGIC-FORCE <=> PTRANS OBJECT GROUND MANNER CYCLIC)

The second, QUAKE1, is the word sense for the lexical item "EARTHQUAKE." It builds a structure but cannot add new roles.

LAST2:

Part Of Speech:	VERB
New Role Locations:	(DURATION)
Semantic Constraint:	TIME-LENGTH
Syntactic Location:	VERB-OBJECT)

The third word sense is LAST2, which is the verb sense of the word "LAST." It contains information where the new role (DURATION) might be found but builds nothing itself.

Thus each Structure Adding Word Sense in FRUMP's dictionary must have a part of speech which indicates how this word sense must be used in the sentence. In addition, it can have a conceptual structure which is its meaning. It can also have information about the location in the sentence where other role fillers might be found.

The second type of word sense, the Role Filling Word Sense, resolves directly to a conceptual token. A conceptual token is denoted by a word of all capitals with asterisks on either side. From now on we will adhere to the asterisk convention for naming memory tokens. The primitive acts are permanent memory tokens. Thus from now on PTRANS will be *PTRANS*, ATRANS will be *ATRANS*, etc. A conceptual token in FRUMP's memory is the location of information about an object. For example, the only word sense of the lexical item "France" resolves to *FRANCE*. *FRANCE* is the conceptual token where all of FRUMP's information about France is stored (e.g. that it is a country). All Role Filling Word Senses also contain their part of speech. As will be seen later, part of speech properties are used by the text analyzer to find syntactic locations in a sentence. An example of a Role Filling Word Sense is FRANCE1:

FRANCE1:
Part Of Speech: NOUN

Conceptual Entry: *FRANCE*

Role Filling Word Senses are much simpler than Structure Adding Word Senses. Role Filling Senses need only indicate the part of speech and the conceptual item to which the word sense resolves. A Structure Adding Sense, on the other hand, must indicate the conceptual structure that is to be built (which can be quite complicated) and the locations of a number of new roles that the word sense can add.

It was mentioned before that words can have any number of word senses. Associated with each lexical item in FRUMP's dictionary is a list of possible word senses for this word. The word senses in this list can be either Structure Adding or Role Filling. For example, the lexical item "BOMB" has two word senses as far as FRUMP is concerned. The first, BOMB1, is a Role Filling Sense. It resolves to the conceptual token *BOMB* which is known to be

a type of weapon. In this sense "BOMB" is a noun. The second sense, BOMB2, is a verb. It builds a conceptualization that means *BOMB*s are being dropped.

Unlike other parsers developed at Yale (Riesbeck & Schank [1976] and Gershman [1979]) FRUMP does not have any explicit "script-specific" extensions to its vocabulary. These other parsers reorganized the dictionary depending on what script was active. This enabled their parsers, for example, to try the correct word sense of "order" first when the restaurant script was active. Instead of preferring word senses on the basis of which scripts are active, FRUMP prefers one word sense over another on the basis of its top-down predictions. The "ask waitress for" sense of "order" is not selected first by FRUMP in restaurant stories because the restaurant script is active but because that word sense fits one of the predicted events - namely MTRANSing one's desire for food to the waitress.

In a way this is a generalization of the dictionary reorganization performed by the other parsers. Dictionary reorganization is helpful because the events corresponding to certain word senses are more likely in one particular script than another. That is, the events corresponding to these word senses are implicitly predicted by the parser when the script is activated. These implicit predictions are made at the time the script is activated. In FRUMP, however, predictions are made and refined continually. Thus a word sense is disambiguated on the basis of the context at the time the word is seen.

5.3.2 FRUMP's Permanent Token Memory

As mentioned previously, the text analyzer only examines words in the context of one or more predictions of what will be found. However, these predictions are often very general. As explained in the previous chapter the PREDICTOR will often only be able to predict characteristics of the desired role filler. The text analyzer must be able to know when it has found something that satisfies the prediction. For example, the PREDICTOR may want a role filled with *VEHICLE*. The conceptual token for any kind of vehicle will do. If the text analyzer finds the word "Chevy," it must know that it has succeeded. Thus it must know that a Chevy is a type of vehicle. The text analyzer must include a general mechanism for answering the question "can the lexical item X be interpreted as a conceptual token Y?". This capability is provided by the organization of FRUMP's permanent token memory.

FRUMP's permanent conceptual tokens (like *CHEVY*, *BOMB*, and *VEHICLE*) are organized hierarchically. For the most part the tokens are arranged in an ISA hierarchy, which has been well described in the literature (Quillian [1968], Raphael [1968], Scragg [1976], and Simmons [1973]). Thus *CHEVY* ISA *AUTOMOBILE* which ISA *VEHICLE* which in turn ISA *PHYSOBJ*.

When the text analyzer is asked to find a *VEHICLE*, it looks at the words in the sentence location where it expects to find the vehicle for a word with a Role Filling Word Sense that resolves to something that can be considered a *VEHICLE*. Each word sense must be checked to see if it can inherit the *VEHICLE* token via its ISA link. However, this is a very easy test and very little computation is expended checking irrelevant words.

If such a word sense is found, its conceptual token is used to satisfy the prediction. For example, if the text analyzer were asked to fill the conceptual role OBJECT with *VEHICLE*, and the word found at the expected sentence location were "Chevy," the text analyzer would add the following structure to the current conceptualization being built:

OBJECT *CHEVY*

5.3.3 FRUMP's Parsing Rules

The text analyzer contains four rules that enable it to interpret the input. Recall that when the SUBSTANTIATOR selection mechanism asks the text analyzer to fill a role, the text analyzer is given the name of the role to be filled along with constraints on what might fill it. Each rule states a strategy for adding a new role to the current conceptualization. The four rules are also rated as to how certain it is that the answer each produces, if it produces one at all, will be correct. There is also a cost estimate of the computing resources needed to apply the rule.

Rule 1:
Find a Structure Adding Word Sense that has been previously processed and can predict the syntactic location of the desired role.
Certainty = 10
Cost = 1

This rule states that one way to add a role is by having an already processed word sense that can suggest where the desired role might be found. For example, suppose FRUMP is processing the sentence

John went to Boston.

and has already chosen the word sense GO1 for the word "went." The conceptualization built thus far will be the conceptualization that GO1 builds:

<=> *PTRANS*

Now suppose that the PREDICTOR asks that the ACTOR role be filled with something that can be considered a kind of *HUMAN*, and the selection mechanism decides that the text analyzer should try to add the role. Recall that the word sense of GO1 looks like this:

GO1:

Part Of Speech:	VERB
Conceptualization Built:	(<=> *PTRANS*)
New Role Locations:	((ACTOR) (OBJECT))
Semantic Constraint:	*ANIMATE*
Syntactic Location:	SUBJECT

When the request for a *HUMAN* in the ACTOR role is given to the text analyzer, it looks through the word senses that have already been processed in building the current conceptualization. If one of them has an ACTOR as a New Role Location, then this rule is applicable. In our example, GO1 can say where the ACTOR role will be found: GO1 thinks it will be in the syntactic subject of the verb. However, to be valid for this reading of the word "go," the subject must be a type of *ANIMATE*.

The text analyzer first checks that the predicted filler is consistent with what the word sense needs. Here, the predicted filler is *HUMAN* and GO1 needs *ANIMATE*. Since *HUMAN* inherits the *ANIMATE* token, everything is fine. If the predicted filler did not inherit the semantic constraint from the word sense, then this word sense would not be able to fill the prediction.

Next, the text analyzer goes to where it expects to find the subject of "go" and finds "John." "John" is in FRUMP's dictionary as a masculine first name. As such, it is known to refer to a *HUMAN* male whose first name is "John." Thus, a filler has been found that satisfies the semantic prediction. The text analyzer then creates a new token for this particular John, and adds the new token to

the current conceptualization as the filler of the both the ACTOR and OBJECT roles. The current conceptualization has been augmented to this:

```
(ACTOR  JOHN1
 <=>    *PTRANS*
 OBJECT JOHN1)
```

This rule is marked with a certainty of 10 on a scale from 1 to 10. This scale does not represent the expected certainty of success in applying the rule. Rather, it is a conditional certainty. Given that the rule produces an acceptable filler, the certainty indicates how likely it is to be the correct filler for the role. For this rule the certainty is 10, the highest, because this rule can succeed only if two constraints are met. First, the text analyzer must predict the correct location in the sentence of the desired role. Second, the text analyzer must find something in that location that satisfies both the requirements of the word sense that predicted the location and the semantic constraints on the filler from PREDICTOR's prediction. In general, parsing rules tend to be very certain compared to inference rules. Later when the role inferencer is discussed it will be seen that roles can often be filled with less certainty.

The certainty of a role filler is attached to it in the conceptualization. This way if a more certain filler is later found for that role the previous one can be replaced.

The cost of applying a rule is a normalized estimation of the computation required to apply that rule. The cost is an integer greater or equal to 1. A higher number indicates more computing resources will probably be spent in applying the rule. The cost is used by the selection mechanism which will be discussed later. The cost of this rule is 1 because it is not very expensive to apply compared to the other rules.

```
Rule 2:
    Find a previously unprocessed word within the
    current phrase that has a word sense that can
    build the desired role and filler.
    Certainty = 10
    Cost = 2
```

This rule gives another way to add a desired role to the current conceptualization. It says that the text analyzer can add a role and filler to a conceptualization by finding a Structure Adding Word Sense that includes the desired role and filler as part of the conceptualization it builds.

For example, consider the sentence

> Israel released a statement condemning Egypt's peace plan.

This input might be seen in an article about a news conference of a head of state. The sketchy script for these news conferences includes among its important events representations for the following two events: one country saying something good about another country, and one country saying something bad about another country. The representations for these events are:

```
(ACTOR      *COUNTRY*
 <=>        *MTRANS*
 MOBJECT    *CONCEPTS*
   TYPE       *APPROVING*
   TOPIC      *CD*
 FROM       *COUNTRY*)
```

and

```
(ACTOR      *COUNTRY*
 <=>        *MTRANS*
 MOBJECT    *CONCEPTS*
   TYPE       *CRITICAL*
   TOPIC      *CD*
 FROM       *COUNTRY*)
```

Something must be said about the MOBJECT TYPE roles here. In the course of its processing, it is necessary for FRUMP to characterize the conceptualization used to fill the MOBJECT TOPIC role. These characterizations are often provided by an inference rule. That is, FRUMP must infer

that a particular event is unfriendly to a certain country. However, the characterizations can on occasion also be provided directly from the text. Since these characterizations might be examined more than once, it is inefficient to have to re-infer them whenever they are needed. So instead, FRUMP incorporates them into the representation.

After "released a statement" has been processed, the PREDICTOR will ask that the MOBJECT TYPE role be filled with either *CRITICAL* or *APPROVING* so that it can select which of the two above representations is to be satisfied.

After "released a statement" has been processed, the current conceptualization is:

```
(<=>     *MTRANS*

 MOBJECT *CONCEPTS*)
```

Now the text analyzer is asked to fill the MOBJECT TYPE role with either *CRITICAL* or *APPROVING*. None of the processed words can successfully tell the text analyzer where the MOBJECT TYPE role will be found in the sentence; rule 1 cannot be used so rule 2 is tried. Rule 2 tells the text analyzer that some other word might be found which includes the desired role in the conceptualization it builds.

In fact, such a word is present in the input. One of the word senses of "condemning" is that it is the gerund form of the verb "condemn." "Condemn" has CONDEMN1 as a Structure Adding Word Sense:

```
CONDEMN1:
    Part Of Speech:              VERB

    Structure Built:             (<=>        *MTRANS*
                                 MOBJECT     *CONCEPTS*
                                   TYPE        *CRITICAL*)

    New Role Locations:          (MOBJECT TOPIC)

          Semantic Constraint     *ACTION*

          Syntactic Location      VERB OBJECT
```

Thus CONDEMN1 can fill the MOBJECT TYPE role with *CRITICAL*.

To apply rule 2 the text analyzer first looks through the sentence for an input word that can be interpreted in a way to add the desired role. The word "condemning" is able to add the MOBJECT TYPE by interpreting it as CONDEMN1. However, the role cannot be added yet. The reason is that there could be conflicts in interpreting "condemning" in this way. For example, if CONDEMN1 built *PTRANS* in the <=> role instead of *MTRANS* it would be unacceptable as a reading for "condemning" because it would conflict with the current conceptualization built thus far. Thus the text analyzer must check that the structure built by the new word sense does not conflict with the structure that already exists. There must also be a low level syntactic check that the word falls within the phrase that built the original conceptualization.

After the conceptualization built by the new word sense has been found not to conflict with the current conceptualization, and the syntactic check has been passed, the new role filler can be added to the current conceptualization. At this time, any other roles present in the word sense conceptualization are also added to the current conceptualization.

Since CONDEMN1 passes both the conceptual match and the syntactic check, the conceptualization it builds is added to the current conceptualization. The current conceptualization now looks like this:

```
(<=>       *MTRANS*

 MOBJECT   *CONCEPTS*

    TYPE      *CRITICAL*)
```

Rule 2 has a certainty of 10 because rather stringent conceptual and semantic tests must be passed before the rule succeeds. Not only must a word have a reading which can build the desired role and filler, but the rest of the structure built must be consonant with the existing conceptual structure. Furthermore, a syntactic test must justify that the word found can legitimately be part of the phrase that built the existing conceptual structure.

The cost of applying this rule is given a rating of 2. This is an ad hoc and somewhat arbitrary indication of how expensive the rule is compared to other rules. It is more costly than rule 1 because it entails searching through the text for the a desired word. In rule 1 there was a prediction of where the desired word would be found in the sentence. No such prediction is made in rule 2.

Rule 3:
Find a previously unprocessed word which has a word sense that can add the desired role as one of its new roles.
Certainty = 10
Cost = 3

This rule provides a method to add a role filler event if the previous two rules fail. If no previously processed word can predict where the filler is, and no unprocessed word can build the filler, this rule says that there might be an unprocessed word that can predict the location of the filler.

For example, consider the following sentence:

Vance met with Gromyko to discuss SALT.

This sentence describes an event in the sketchy script $VIP-MEET. The representation for that event is:

```
(ACTOR          *GROUP*
   MEMBER         *VIP*

 <=>            *MTRANS*

 MOBJECT        *CONCEPTS*
   TOPIC          *CD* involving
                       countries of *VIP*

 FROM           *GROUP*
   MEMBER         *VIP*

 TO             *GROUP*
   MEMBER         *VIP*)
```

This conceptualization says to expect a mental transfer of concepts among a group of VIPs concerning some conceptual structure. Part of this predicted event will be built from the phrase "Vance met with Gromyko". After processing "Vance met with Gromyko," the current conceptualization will look like this:

```
 (ACTOR          *GROUP*
    MEMBER         (*VANCE* *GROMYKO*)

  <=>           *MTRANS*

  MOBJECT       *CONCEPTS*

  FROM          *GROUP*
    MEMBER         (*VANCE* *GROMYKO*)
```

```
TO              *GROUP*
   MEMBER         (*VANCE* *GROMYKO*))
```

This conceptualization matches the predicted conceptualization except that the MOBJECT TOPIC role is missing. At this point the PREDICTOR will ask that the MOBJECT TOPIC be filled with a conceptualization involving the U.S. and Russia. It predicts that the U.S. and Russia will be involved because those are the countries that the VIP's represent.

The following is the word sense of "met" that has been assigned by the text analyzer:

```
MEET1:
     Part Of Speech:          VERB

     Structure Built:        (<=>     *MTRANS*
                              MOBJECT *CONCEPTS*)

     New Role Locations:      (MOBJECT TOPIC)
        Semantic Constraint:    *CD*
        Syntactic Location:     (PREP-OBJECT about)

     New Role Locations:      ((ACTOR) (TO) (FROM))
        Semantic Constraint:    (*GROUP*
                                   MEMBER *HUMAN*)
        Syntactic Location:     ((SUBJECT)
                                   (PREP-OBJECT with))
```

The current conceptualization has been built from the structure MEET1 builds and by applying rule 1 to add the ACTOR, TO, and FROM roles from where MEET1 predicted them.

MEET1 also predicts where the MOBJECT TOPIC role will be found. It predicts it will be the object of the preposition "about." The word sense must have this information if the text analyzer is to process a sentence like "Vance met with Gromyko about SALT." However, in our example sentence, there is no preposition "about." Thus the MOBJECT TOPIC role cannot be filled by rule 1. Nor can rule 2 help; no text word can directly build the MOBJECT TOPIC role.

Instead, rule 3 is applied. Rule 3 says to find another word in the sentence that has a word sense that can predict where the desired filler will be. The word "discuss" is found. "Discuss" has a word sense DISCUSS1 which looks like this:

```
DISCUSS1:
    Part Of Speech:          VERB

    Structure Built:        (<=>      *MTRANS*
                             MOBJECT *CONCEPTS*)

    New Role Locations:     (MOBJECT TOPIC)
       Semantic Constraint:    *CD*
       Syntactic Location:     VERB-OBJECT

    New Role Locations:     ((ACTOR) (TO) (FROM))
       Semantic Constraint:    (*GROUP*
                                  MEMBER *HUMAN*)
       Syntactic Location:     ((SUBJECT) (PREP-OBJECT with))
```

DISCUSS1 looks very much like MEET1 except it predicts that the MOBJECT TOPIC role will be found as its syntactic object. Before this prediction can be used, however, the text analyzer must make sure that DISCUSS1 can pass the same semantic and syntactic constraints as were needed in rule 2. That is, the conceptual structure built by the word sense must not conflict with the conceptual structure already present in the current conceptualization. Furthermore, the new word must be in the same syntactic phrase as the words that built the current conceptualization.

In our example, DISCUSS1 passes both of these tests. The conceptualization built by the word sense is the same as the conceptualization built initially by "met," and the word "discuss" is in the same phrase as "met." Thus the text analyzer knows to look in the object of the verb "discuss" for the filler of the MOBJECT TOPIC role. It also expects that the filler will be a conceptualization involving Russia and the U.S. This is from PREDICTOR's original prediction.

In the location of the syntactic object of the verb "discuss," the text analyzer finds "SALT." "SALT" is in FRUMP's dictionary and has a Role Filling Word Sense SALT2. SALT2 resolves to the conceptual token *SALT-TREATY* which is a type of *AGREEMENT* and involves Russia and the U.S. Thus, in the word "SALT," the text analyzer finds just what the PREDICTOR wanted as a filler of the MOBJECT TOPIC role. The role is then added to the current conceptualization to make:

```
(ACTOR          *GROUP*
   MEMBER         (*VANCE* *GROMYKO*)

 <=>            *MTRANS*

 MOBJECT        *CONCEPTS*
   TOPIC          *SALT-TREATY*

 FROM           *GROUP*
   MEMBER         (*VANCE* *GROMYKO*)

 TO             *GROUP*
   MEMBER         (*VANCE* *GROMYKO*))
```

The certainty of this rule is 10. If this rule produces the filler of a desired role, that filler must have been found in the predicted syntactic location. Furthermore, the word sense that predicted where the role would be found must have built a structure consonant with the existing structure and must have occurred in the same syntactic phrase.

The expected cost of the third rule is 3. It is more expensive to apply this rule than either rule 1 or rule 2 because it requires looking at the text twice. The text must be searched once looking for the unprocessed word whose word sense can predict the location of the desired filler, and once to actually find the filler.

Rule 4:

Find a word regardless of any syntactic considerations that has the desired semantic properties.
Certainty = 3
Cost = 1

This rule provides a quick and dirty, though uncertain, way to guess at a role filler if the text analyzer cannot make sense of the syntax of the input sentence. Essentially it says "never mind about syntax, find any word that can resolve to a token that satisfies the semantic constraint of the PREDICTOR." Rule 4 allows the text analyzer to produce a guess at an interpretation of a text even if all its other rules fail.

For example, suppose FRUMP is given the following sentence:

Israel and Egypt reached an agreement today on a new treaty.

But suppose that "reached" is not in FRUMP's vocabulary. Rule 4 will allow the text analyzer to correctly process the sentence even though it no longer makes sense syntactically. To FRUMP the sentence now will look like the following:

> Israel and Egypt XXXXXXX an agreement today on a new treaty.

Of course, FRUMP still must know the word "agreement." FRUMP's dictionary definition of "agreement" states that it is the nominalized form of the verb sense AGREE1. The fact that it is a nominalized verb tells the text analyzer two things: 1) the word behaves syntactically as a noun (which is unimportant for this example) and 2) that the new role predictions concerning the SUBJECT and VERB-OBJECT locations of the phrase are no longer meaningful. The word sense AGREE1 looks like this:

```
AGREE1:
    Part Of Speech:              VERB

    Conceptualization Built:     (<=>          *MTRANS*
                                  MOBJECT      *CONCEPTS*
                                     TYPE         *APPROVING*

    New Role Locations:          ((ACTOR) (FROM) (TO))
        Semantic Constraint:     *ANIMATE*
        Syntactic Location:      ((SUBJECT)
                                  (PREP-OBJECT with))

    New Role Locations:          (MOBJECT TOPIC)
        Semantic Constraint:     *CD*
        Syntactic Location:      ((PREP-OBJECT on)
                                  (PREP-OBJECT about))
```

For the nominalized verb "agreement" everything in the dictionary definition for AGREE1 is valid except the prediction that the ACTOR, TO, and FROM roles will be found in the syntactic subject location.

The input sentence might be seen in the context of a story about negotiations. FRUMP has a sketchy script, $NEGOTIATION, for that situation. The representation of the expected event that the input text must match is the following:

```
(ACTOR          *GROUP*
   MEMBER          (*COUNTRY*)

 <=>            *MTRANS*
```

```
MOBJECT          *CONCEPTS*
   TYPE             *APPROVING*
   TOPIC            *CD*

FROM             *GROUP*
   MEMBER           (*COUNTRY*)

TO               *GROUP*
   MEMBER           (*COUNTRY*))
```

This says that a group of countries all approve of a particular topic.

After the word "agreement" has been processed, the current conceptualization is the conceptualization built by AGREE1:

```
(<=>        *MTRANS*

   MOBJECT  *CONCEPTS*
      TYPE     *APPROVING*)
```

The problem that parsing rule 4 will help to solve is finding the countries to fill the ACTOR, TO, and FROM roles. AGREE1 predicts that these roles might be filled by the object of the preposition "with." However, there is no "with" in the example input. The prediction that the roles will be found in the syntactic subject location are, of course, not useful since AGREE1 came from a nominalized verb.

If FRUMP knew the word "reached," it could be used to predict that its syntactic subject would be the countries. However, since the word is not known, the text analyzer can have no way of predicting the syntactic location of the countries involved.

At some point in the processing, the PREDICTOR will ask that the ACTOR role be filled with a group of countries. This prediction comes directly from the negotiations script. The script event dictates that the ACTOR must be filled with a group of countries. When this prediction is given to the text analyzer, it tries rules 1 - 3. They all fail since no word in the input can successfully predict the syntactic location of the ACTOR role. Rule 4 is then tried. It simply looks for a group of countries in any syntactic position. The only group of countries in the sentence is from the string "Israel and Egypt." Parse rule 4 says to take these countries and add them to the current conceptualization in the ACTOR role but with a low certainty. AGREE1 contains the information that the ACTOR, TO, and FROM roles all have the same filler. Thus, after parse rule 4 is tried, the current conceptualization looks

like this:

```
(ACTOR          *GROUP*
   MEMBER         (*ISRAEL* *EGYPT*)

 <=>            *MTRANS*

 MOBJECT        *CONCEPTS*
   TYPE           *APPROVING*

 FROM           *GROUP*
   MEMBER         (*ISRAEL* *EGYPT*)

 TO             *GROUP*
   MEMBER         (*ISRAEL* *EGYPT*))
```

The MOBJECT TOPIC role is added in the normal way via rule 1 and AGREE1. Since the word "treaty" is specified as the object of the preposition "on," the prediction of AGREE1 of the syntactic location of MOBJECT TOPIC role is still usable.

Rule 4 has a low certainty of 3 because there is no syntactic evidence for adding the role. It is added only on semantic grounds. The certainty is attached to each of the roles added. Thus from parse rule 4, the ACTOR, TO, and FROM roles all have a certainty of 3. In this way, if a role can be added later with a higher certainty (either by another parse rule or an inference rule) the new filler can be used to replace the less certain old one.

The cost of rule 4 is very low. It is given a cost of 1 since comparatively little work need be done to apply the rule. There is no syntactic work at all. The only processing needed is to find a word or phrase in the sentence with the desired semantic property.

5.3.4 Syntax

FRUMP's understanding is primarily driven by semantic considerations. Syntactic information is, however, used by the text analyzer to provide clues about where in a sentence a desired conceptual role filler might be found. That is, it mediates between conceptual roles and sentence locations. In this way syntax can limit searching through the text. If, for example, the text analyzer has decided that the role filler it is trying to add will be found in the syntactic subject, it need only consider nouns in front of the verb as candidates. Since this greatly reduces the number of words to be considered, the process is made more efficient.

Consider, for example, how the following sentence would be processed without syntactic information:

Israel invaded Egypt.

The word "invaded" will build some structure enabling the PREDICTOR to request that the ACTOR role be filled with a country. Without syntactic information, the text analyzer will not be able to predict that the ACTOR will be found in the subject location. In fact, "the subject location" will have no meaning. There are two countries mentioned in the sentence: Israel and Egypt. Without further information, there is no way to prefer one over the other as the filler of the ACTOR role.

With syntactic knowledge, however, the ambiguity disappears. The text analyzer can make the prediction that the ACTOR role will be found as the subject of the verb. It also knows that the subject is a noun in front of the verb. This information allows the text analyzer to use "Israel" as the word that fills the ACTOR role.

The text analyzer has a very incomplete knowledge of syntax. It knows about the subject-verb-object construction of English sentences, it knows roughly where to look in the sentence for various syntactic locations, and it knows that complex sentences can be built up from simple clauses.

Having only incomplete syntactic knowledge is, surprisingly, an advantage for FRUMP. The alternative is to give FRUMP an explicit grammar like many other natural language systems (Marcus [1977], Winograd [1972], and Woods & Kaplan [1971]). There are two disadvantages to having such a grammar. First, the grammar would have to be very complicated if it were to account for a large part of English. This would make processing much less efficient. Second, English has eluded every attempt at constructing a rigorous grammar for it. No matter how complicated the grammar, there would be large classes of English sentences that could not be parsed. Thus an explicit grammar is overly constraining.

The text analyzer has 9 syntactic rules which enable it to find the correct sentence locations in the text that correspond to syntactic labels (e.g. SUBJECT, OBJECT OF PREPOSITION, etc.). The first 4 rules tell the text analyzer where to find the syntactic labels. The last 5 indicate how the sentence structure is changed by passive and nominalized verbs.

Syntax Rule 1:
The SUBJECT of a verb is a noun preceding the verb which is not immediately preceded by a

preposition.

This rule enables the text analyzer to locate the syntactic subject of a verb. The text analyzer can only reject as candidates for the subject nouns that are immediately preceded by a preposition, like "to John." This is because any unknown or unprocessed intervening words might terminate the prepositional phrase, for example, "To the zebra John gave an apple." If the rule did not include "immediately preceded" and the word "zebra" were unknown, the rule would not allow "John" to be the subject of the sentence since it would be treated as the object of the preposition. This rule, like all of FRUMP's syntax rules, is only a heuristic. As such it should help processing where possible, but it need not always work. There is room for improvement in all of FRUMP's syntax rules. However, in their current form they are quite adequate for their job.

The syntax rules should never prevent the correct interpretation of the text. Stated in the way it is, rule 1 lets too much through. However, the system can survive this deficiency. Incorrect possibilities might be ruled out on semantic grounds. If, on the other hand, the rule eliminated at this low level the actual subject of certain sentences, the system could never recover. Articles, however, are not included as separators so in the phrase "to the boy," the word "boy" could not be the subject.

To illustrate this rule, consider the sentence

The boy went.

Suppose that the only processing done so far is to resolve the word "went" to the word sense GO1. Now suppose the PREDICTOR requests that the conceptual ACTOR role be filled with something that can be considered a *HUMAN*. The word sense GO1 contains the information that the conceptual ACTOR role will be found in the syntactic subject of the verb. This rule enables the text analyzer to find the approximate location in the sentence of the desired word. The text analyzer looks backwards from the verb "went" for a noun that is not immediately preceded by a preposition. The word "boy", which has a word sense that satisfies these syntactic constraints, is found. The text analyzer then checks whether that word sense satisfies the semantic requirements (i.e. that it can be a type of *HUMAN*). It does and so *BOY*, the conceptual referent of the selected word sense of "boy" is used to fill the ACTOR role.

Syntax Rule 2:
The VERB-OBJECT of a verb is a noun following the verb which is not immediately preceded by

a preposition.

The text analyzer does not distinguish between direct and indirect verb objects. Both of these syntactic types are considered VERB-OBJECTs.

The reason for not differentiating between direct and indirect objects is that the indirect object location is not well defined until the direct object of the verb has been found. The indirect object is a noun which is not the object of a preposition whose location is between the verb and the direct object. Since the location of the indirect object depends on the location of the direct object, it cannot be determined until the direct object has been resolved.

This is a problem, since the location of the direct object is not resolved until it is used to substantiate a prediction from PREDICTOR. Thus the indirect object location is not well defined until some prediction is filled using the direct object. To use the indirect object location the text analyzer must have previously used the direct object location. That is, the prediction using the syntactic direct object must be made before the prediction using the indirect object. This is too confining a constraint to place on the PREDICTOR. Instead, the text analyzer simply distinguish between the direct and indirect objects as in terms of sentence location.

This means that there are some sentences FRUMP cannot in principle process (i.e. those with semantically similar direct and indirect objects). However, this has not proved to be a serious constraint to FRUMP's performance. If it becomes a problem in some future domain, syntax rule 2 will have to be improved. This will be an added inconvenience to the PREDICTOR but is not at all impossible. As yet, however, the expected benefit is not worth the expense.

Syntax Rule 3:
The PREP-OBJECT of a preposition is a noun following the preposition with no intervening prepositions.

This rule enables the text analyzer to find objects of prepositions. Consider the sentence

An automobile crashed into a billboard.

The word "billboard" must be added to the conceptualization being built. In this sentence the selected word sense of "crashed" builds a *PROPEL*. Further processing results in

the ACTOR role being filled with *AUTOMOBILE*. Using the vehicle accident sketchy script, the PREDICTOR requests that the conceptual OBJECT be filled with some kind of *PHYSOBJ*. The selected word sense of "crashed" contains the prediction that the conceptual OBJECT of the *PROPEL* will be found as the PREP-OBJECT of "into." This rule tells the text analyzer how to find the PREP-OBJECT of "into." It says to look after the preposition for a noun. In this case the text analyzer finds the word "billboard" which can be considered a kind of *PHYSOBJ* and so is added to the conceptualization as the filler of the OBJECT role.

Syntax Rule 4:
The MODIFIER of a noun is a noun or adjective preceding the dominating noun.

This rule enables the text analyzer to find information typically found as a modifier. For example, consider the sentence

England seized an Icelandic trawler.

This sentence must be represented as a change of possession from Iceland to England. That is, it will be an *ATRANS* with the conceptual FROM role filled with *ICELAND*. The sentence does not explicitly say from whom the trawler was taken. Rather it specifies the owner of the trawler. To fill in the FROM role with *ICELAND* FRUMP must know that a change in possession of an object is FROM its previous owner. Furthermore, the text analyzer knows that ownership is typically specified in English by a possessive adjective modifying the owned noun. This syntax rule enables the text analyzer to find that Iceland is the owner of the trawler. The seize sketchy script requires a *COUNTRY* to fill the FROM role. FRUMP figures out that the FROM role will be filled with the owner of the trawler. The PREDICTOR then asks that owner of the trawler be found with the constraint that it must be a country. The text analyzer can use this syntax rule to look backwards from the word "trawler" until it finds a country that can be interpreted as the owner. It finds "Icelandic" which is then used to fill the desired role.

Syntax Rule 5:
When the past participle of a verb is found and the verb is preceded by a form of the verb "to be," assume the verb is passive.

This rule is simply the way FRUMP recognizes passive verbs. It is necessary to recognize passives because the syntactic locations of SUBJECT and VERB-OBJECT are altered.

The next two rules indicate to FRUMP how these locations are changed.

Syntax Rule 6:
When looking for the SUBJECT of a passive verb, look instead for the PREP-OBJECT of "by."

Syntax Rule 7:
When looking for the VERB-OBJECT of a passive verb, first look in the SUBJECT locations. If no word can be found with the desired properties, look in the VERB-OBJECT location.

Rules 6 and 7 tell the text analyzer how to modify where it looks in a phrase if the verb is passive. The passive rule is that one of the objects of the verb (either direct or indirect) becomes the subject, and the subject optionally becomes the object of the preposition "by."

Nothing special has to be done to handle the fact that the subject is optionally moved to the object of "by." If no satisfactory object of the preposition "by" is found, the text analyzer will simply fail to add the role. The filler will then have to be found by other means.

Rule 7 specifies how the VERB-OBJECT is changed. In English, there are two possible ways to form the passive. In one, the direct object of the verb is moved to the subject position. For example,

1) John gave Mary the present.

can become

2) The present was given Mary by John.

The second way is to move the indirect object to the subject position. In this case sentence (1) becomes

3) Mary was given the present by John.

The result of these two ways to form the passive is that when the text analyzer is looking for VERB-OBJECT of a passive phrase, it might either be found in the SUBJECT location or the VERB-OBJECT location. Thus Rule 7 must check both places.

Syntax Rule 8
When looking for the SUBJECT of a gerund or nominalized verb, look instead for the PREP-OBJECT of "by."

Syntax Rule 9

When looking for the VERB-OBJECT of a gerund or nominalized verb, look instead for the PREP-OBJECT of "of."

A nominalized verb is a verb made into a noun, typically by adding the ending "tion." For example, the verb "nationalize" may be transformed into the noun "nationalization." The word sense for "nationalize" is:

```
NATIONALIZE1
        Part Of Speech:                 VERB

        Conceptualization Built:        (<=>        *ATRANS*
                                         OBJECT     *CONT*
                                           TYPE       *ECONOMIC*)

        New Role Locations:             ((ACTOR) (TO))
                Semantic Constraint:     *POLITY*
                Syntactic Location:      SUBJECT

                                        (OBJECT)
                Semantic Constraint:     *SPEC-INDUSTRY*
                Syntactic Location:      VERB-OBJECT
```

The words "nationalize" and "nationalization" have very similar meanings. FRUMP does not have morphological decomposition rules. However it is very useful to be able to use the meaning of the verb as the meaning of the noun. This way there will be no need to duplicate most of the dictionary information. The nominalized definition can simply have a pointer to the verb definition together with the information that it is used as a noun. However, some of the new role locations of the noun's word sense are different from the verb's word sense. For example, as a verb "nationalize" can be used as follows:

Uganda nationalized a French company.

However, the noun "nationalization" can express the same information as

The nationalization of a French company by Uganda...

Rules 8 and 9 permit the text analyzer to make this transformation. The word sense of "nationalization" can now simply be a pointer to the word sense for the verb "nationalize" together with an indication that the verb has been nominalized:

NATIONALIZATION1:
Part Of Speech: NOUN

Root: (NOMINALIZE NATIONALIZE1)

The definition of "nationalization" is then very much more compact than it would be if all of the information were to be duplicated from the "nationalize" entry.

Thus, instead of an explicit grammar, FRUMP relies on its top-down predictions and a set of simple heuristic rules about English syntax. These rules are far from a complete specification of English syntax and will occasionally make mistaken predictions about where in the sentence to look. However, FRUMP's incomplete knowledge of syntax allows the advantages of constraining the search for desired words and eliminating certain semantic ambiguities. Since all of the processing is prediction driven, the text analyzer retains a flexibility beyond any practical syntactic grammar. The text analyzer's syntactic knowledge finds the general sentence location of the desired word while the semantic predictions determine exactly which word and word sense will be used.

In summary, FRUMP has nine syntactic rules. Ultimately more will probably be needed if FRUMP is extended to do more detailed processing. However, FRUMP's syntactic heuristics so far are quite adequate. The text analyzer uses its syntactic knowledge to constrain where to look in a text phrase for a desired conceptual item. That is, once the text analyzer is asked to fill a conceptual role, it applies its syntactic rules to determine where in the input it ought to look for the word that will provide the role filler. The syntax rules are only applied when needed. The system does not do a syntactic parse of each input sentence. Instead it builds only as much of a syntactic parse tree as it needs to build the conceptual representation of the sentence. The parse tree is augmented under the direction of the conceptual processing.

5.3.5 Anaphoric Reference

FRUMP's top-down orientation allows the text analyzer to resolve a large class of pronominal references easily. Recall from the discussion of the PREDICTOR that its predictions are constantly revised to be the tightest, most accurate possible. At times the PREDICTOR can anticipate the precise filler of a desired role. That is, instead of predicting that the ACTOR role will be filled with just

COUNTRY, it can predict that the country will be *FRANCE*. This type of prediction comes from PREDICTOR rule 9.

Recall that PREDICTOR rule 9 is:

PREDICTOR rule 9
> If a role filler is predicted, and that role is filled by a previously bound script variable, predict the more explicit binding of the script variable instead of the less specific filler constraint from the predicted conceptualization.

This kind of prediction makes resolving a pronoun particularly easy. The text analyzer responds to this type of prediction in the normal way. It looks in the syntactic location for something that will resolve to the predicted token (e.g. a word that can mean *FRANCE*). If it finds a pronoun that does not conflict with properties of the predicted token, the predicted token is used to fill the role.

Of course, there is the possibility that the pronoun will conflict with properties of the predicted conceptual item. For example, if *JOHN* were predicted and the pronoun "she" found in the text. In these cases, the text analyzer must look for some other word that has a reading corresponding to *JOHN*. If such a word is found, the corresponding conceptual item is added to the current conceptualization. If not, then the text analyzer can not add the desired role, and it informs the selection mechanism of that fact so that another method might be tried.

Recall from the previous chapter that PREDICTOR rule 11 is:

Rule 11
> If a single explicit role filler cannot be decided upon, predict that the role will be filled from the list of possible explicit role fillers.

There is the possibility that PREDICTOR used rule 11 instead of rule 9 to make the prediction. Rule 11 predicts a list of explicit items that might be the role filler. For example, the text analyzer might be asked to fill the ACTOR role with one of the elements of the list (*JOHN* *BILL* *MARY*). Now, if a pronoun is found in the expected

syntactic location, there could be trouble. If the pronoun "she" is found everything is fine because the properties of "she" conflict with all of the predicted fillers except *MARY*. However, if the pronoun "he" is found, it could match either *JOHN* or *BILL*.

When several predicted items all match a pronoun, the text analyzer resorts to syntax. Obviously, world knowledge cannot be used at this point to disambiguate the pronoun. If there were any semantic reason why one of the items ought not match the pronoun, the PREDICTOR would not have included it in the first place. The only alternative is the use of syntactic knowledge.

The text analyzer currently has only one syntactic rule to aid in disambiguating pronouns:

> Syntax Rule 10
> prefer a referent if it appeared in the same syntactic location of the previous clause.

To illustrate this rule, consider the following sentences:

> President Carter met today with Vice Premier Teng. He proposed a trade agreement between China and the U.S.

It is quite clear that Carter is the one who made the proposal. However, there is no world knowledge that could disambiguate the pronoun "he" in favor of "Carter." A visiting diplomat is as likely to make a proposal as is the host diplomat. The pronoun "he" is the syntactic subject. "Carter" is the preferred referent simply because "Carter" also appeared as the syntactic subject of the previous sentence.

Of course, this syntactic rule alone is not sufficient to handle all of the semantically ambiguous pronouns that could occur. However, if it were a problem it would be easy to add more such syntactic rules. Semantically ambiguous pronouns are not very common. If there is any question as to the correct referent, the writer of the news story will not use a pronoun. Of the semantically ambiguous pronouns, this single syntactic rule is able to resolve a large number of them. There is little need for more sophisticated syntactic rules here.

Deictic references are widely used in news stories to refer to dateline information. For example, consider the beginning of a news story:

> Chtaura, Lebanon, Oct. 8 - Syrian, Lebanese and Palestinian representatives will meet here tomorrow to discuss the withdrawal of Palestinian forces.

In this story "here" and "tomorrow" must be interpreted with respect to the dateline information. "Here" must resolve to Chtaura, Lebanon, and "tomorrow" must resolve to October 9, 1975.

These deictic references are relatively straightforward to handle. FRUMP processes the dateline information in the header of UPI stories. This establishes both the time and the place where the story originated. Time and place pronouns encountered with no explicit conceptual prediction are interpreted as referring to this dateline information. The dictionary definition for relative pronouns like "tomorrow," include how they modify the referent (e.g. "tomorrow" is the referent date plus one day).

In FRUMP's method of processing, pronouns are not a problem. In fact, they are an advantage. Pronouns improve processing efficiency because they eliminate an inheritance check that would otherwise be performed. In many systems pronouns are handled differently than other processing. When a pronoun is encountered, a special routine is called to find a referent for it (although Charniak argued in his dissertation [1972] against this technique).

In FRUMP, however, due to the nature of its processing, referents are predicted in the same top-down manner as everything else. When a pronoun is encountered, under normal circumstances, FRUMP's PREDICTOR will already have anticipated an explicit filler for the desired role.

For example, consider the pronoun occurring in the second sentence of the following input:

> Uganda nationalized an Exxon oil refinery. It was paid $1.2 million in compensation.

After processing the first sentence, FRUMP will realize that this is a story about one country nationalizing a company of another. The first sentence builds a meaning representation in which there is a forced *ATRANS* of an oil refinery. The ACTOR is *UGANDA*, the conceptual referent for "Uganda." The FROM role is filled with *EXXON*, the conceptual referent for the word "Exxon." These fillers are bound to script variables for the entity taking the industry and the entity giving it up, respectively. In the second sentence, FRUMP will build the structure

(<=> (*ATRANS*)

```
OBJECT    (*MONEY*)
  AMOUNT  (1200000)
  UNIT    (*DOLLAR*))
```

from the phrase "paid $1.2 million." At some point the PREDICTOR will ask that the TO role be filled.

The PREDICTOR can immediately predict that the filler of the TO role will be *EXXON*. This is done because the only script event that the above partial conceptualization matches is the one for giving compensation to the previous owner of the nationalized industry. Furthermore, the script variable for the entity that gave up control of the industry has already been bound to *EXXON* from processing the previous sentence. If this conceptualization is indeed to match the predicted event of paying compensation, the only possible filler of the TO role is *EXXON*. *EXXON* is predicted to be the filler of the TO role by PREDICTOR rule 9 which states that the binding of a script variable should be predicted if it is known.

The SUBSTANTIATOR selection mechanism calls the text analyzer to fill the TO role with something that can be considered *EXXON*. The text analyzer predicts that the TO role will be filled with the syntactic subject (using syntax rule 7 since the verb is passive). In the syntactic subject location the text analyzer finds the pronoun "it." Since a single explicit filler has been predicted with which this pronoun does not conflict, the predicted filler is assumed.

A pronoun in the predicted syntactic location eliminates the need for an inheritance match to be performed. It says basically "never mind the further processing you would do if there were a real word here, the prediction you have is the correct one provided it matches in gender and number." If in the above example, "the Exxon Corporation" were found instead of "it," the text analyzer would have to justify that "the Exxon Corporation" could indeed be considered to refer to *EXXON*. Although in this case the justification would be quite easy, it would involve some work, and in general could involve a good deal of inheritance matching. Thus, using FRUMP type parsing, pronouns make text interpretation more efficient rather than causing problems.

5.3.6 Looking at More Than One Word at a Time

At the beginning of this section it was stated that the text analyzer looks at only one word at a time with three exceptions. The exceptions are 1) passives, 2) phrases, and 3) composites.

Passives are straight forward to handle. When the text analyzer uses a word that might be the past participle of a verb, it looks to see if it is preceded by a form of the verb "to be." If so, FRUMP assumes it is passive. In processing that word, however, the text analyzer needed to look for the auxiliary "to be."

Phrases are groups of words that have a special meaning only in conjunction with each other. Phrases are either idiomatic constructions or, as is more often the case in FRUMP, multi-word names like "the United States." There is a dictionary entry for each phrase. The dictionary entry of the phrase is added as a word sense to any one of the words in the phrase (usually the rarest word). The dictionary entry includes instructions on how to justify that the rest of the phrase is present. For example, one of the senses of the word "united" says "if I'm capitalized and I am followed by the word "states" also capitalized then I can resolve to *USA*. When the text analyzer wants to use a phrase definition, it must first justify that the rest of the phrase is indeed present.

Composites are permanent memory tokens for which there is no single lexical realization. For example, there is a node *QUAKE-MAGNITUDE* which is used to bind the &MAGNITUDE variable in the earthquake script. The *QUAKE-MAGNITUDE* memory pointer is composed of two parts: 1) a *QUAKE-SCALE*, and 2) a number. Examples of possible *QUAKE-MAGNITUDE*s might be a Richter scale reading of 4.4, or a Mercali scale reading of 3.7. In both cases it is composed of a scale and a level reading on that scale. Thus when the text analyzer is asked to find a composite memory token, it must build it up from its pieces. In recognizing a composite memory token then, the text analyzer must examine several words.

5.4 The Role Inferencer

The role inferencer is the final SUBSTANTIATOR subsystem that can build conceptual structures. Like the text analyzer, it cannot respond to predictions of entire conceptualizations. Rather it operates on one role at a time.

The role inferencer is composed of a large number of inference rules. Each inference rule contains three parts: the role it can add, conditions on when it applies, and a specification of the filler that can be added. In addition, each inference rule has a certainty and a cost specification similar to the parsing rules. The certainty, again, indicates how sure the system can be of a result produced by

this rule. The cost is an estimation of the expense of applying the rule.

Since there are many inference rules, there must be an efficient method of indexing them. Inference rules are organized by the primitive acts or states of the conceptualization, and within each primitive act or state by the roles that they can fill. Thus, all the rules that can add the conceptual role OBJECT to an *ATRANS* conceptualization are listed together, all the rules that can add the role *ACTOR* to a *PTRANS* conceptualization are listed together, etc. These groups of inferences are sorted high to low by their certainties. This makes the selection procedure more efficient.

For example, even though there are many inference rules in the FRUMP role inferencer, there are only five that might be applicable to inferring the FROM role of *ATRANS* acts. Since the role inferencer always knows the act of the conceptualization it is augmenting, and it is told the role that the PREDICTOR wants filled, it can immediately retrieve the possibly relevant inference rules. If it is trying to infer the FROM role of an *ATRANS*, it can immediately narrow the relevant inference rules to these five. Furthermore, since these five rules are sorted by certainty, it can easily try the most certain rules first.

The conditions of when a rule can apply are a series of tests to be applied to other role fillers of the conceptualization. For example, there is a rule to infer that when planes crash, they usually crash into the ground. This rule is indexed by the primitive act *PROPEL*, and the conceptual role OBJECT because it can fill the OBJECT role of *PROPEL* conceptualizations. The rule is:

Primitive Act:	*PROPEL*
Desired Role:	(OBJECT)
Inferable Filler:	*GROUND*
Tests:	(ACTOR) filled with *AIRCRAFT* (MANNER) filled with *VIOLENT*

This rule states that the OBJECT of a *PROPEL* may be filled with *GROUND* if the filler of the ACTOR role can inherit *AIRCRAFT* and the MANNER can inherit *VIOLENT*.

Of course, not all violent propels of an aircraft need be to the ground. A plane can, for example, crash into another plane or a building. However, if either of these are the case, the object crashed into must be explicitly mentioned in the text. It is the job of the selection

procedure to insure that an inference such as this is not made when there is contrary information in the text.

The certainty of an inference reflects how sure it is of the filler it produces. The certainty of the above inference is 8 on a scale of 1 to 10.

The cost of applying an inference rule is dynamically computed. It is twice the number of unfilled roles that must be examined to apply the rule plus one. This cost function normalizes the cost of an inference to the cost of applying a parsing rule. The average cost of a parsing rule is 2. Thus assuming the parser was able to fill the needed roles, the average cost of filling the roles would be twice the number of missing roles. One is added to account for the cost of manipulating the inference rule. Thus the maximum cost of applying the "crash into ground" rule is 5. This is the cost if neither the ACTOR or the MANNER rule is present in the conceptualization when the OBJECT role is desired by PREDICTOR. The minimum cost is one if both the ACTOR and MANNER roles are already filled in.

The inference rules may call the SUBSTANTIATOR with predictions of their own. This is done if a desired role is missing from a conceptualization. To illustrate this, we will discuss how FRUMP processes the following sentence in the context of the script $VEHICLE-ACCIDENT:

A jet crashed.

The word "crashed" builds the structure

```
(<=>      *PROPEL*
 MANNER   *VIOLENT*)
```

Furthermore, it predicts that the ACTOR will be found in the syntactic subject and that the OBJECT will be found as the object of the preposition "into."

The relevant event in the vehicle accident sketchy script is the following:

```
(ACTOR    *VEHICLE*
 <=>      *PROPEL*
 OBJECT   *PHYSOBJ*
 MANNER   *VIOLENT*)
```

Now suppose PREDICTOR wants the OBJECT filled in next. On

the basis of the sketchy script, it will predict that the filler will be some kind of *PHYSOBJ*. The SUBSTANTIATOR selection procedure will first ask the parser to fill the role. The parser will fail. Exactly why it is called first and why it does not employ parse rule 4 (the one that will take anything) to fill the OBJECT role with *JET* will be discussed in the next section. For now, it is sufficient to know that the parser was called and failed.

The inference rules are tried next. The primitive act of the current conceptualization is *PROPEL*; the desired role is OBJECT. Thus the "crash into ground" inference will be among them. From the group of inferences found, the viable inferences are collected. These are the inferences which have a chance at supplying the desired information and whose test conditions are not violated by roles present in the current conceptualization. Our "crash into ground" inference is a viable inference. The filler it can add, *GROUND*, is indeed a kind of *PHYSOBJ* (so it has a chance of adding the desired filler), and the only non-missing test role (MANNER) is satisfied (filled with *VIOLENT*). At this point in the processing the cost of applying this inference rule is 3. There is one unfilled role (the ACTOR) which needs to be tested.

The role inferencer is allowed to recursively call the SUBSTANTIATOR to fill missing roles that must be tested. In our example, the role inferencer now calls SUBSTANTIATOR to fill the ACTOR with a kind of *AIRCRAFT*. SUBSTANTIATOR again tries the parser first. Via parse rule 1, it finds the word "jet" in the predicted subject location. "Jet" indeed can be interpreted as a kind of *AIRCRAFT* so *JET* is added to the current conceptualization which now looks like this:

```
(ACTOR    *JET*
 <=>      *PROPEL*
 MANNER   *VIOLENT*)
```

Now all of the constraint tests of the inference rule are satisfied. The inference rule fills the role OBJECT with *GROUND* making the following conceptualization:

```
(ACTOR    *JET*
 <=>      *PROPEL*
 OBJECT   *GROUND*
 MANNER   *VIOLENT*)
```

This conceptualization matches the predicted sketchy script event, which PREDICTOR now marks as having been satisfied.

There is another type of inference rule which varies slightly from the form of the "crash into ground" inference. The "crash into ground" inference always supplies the same filler (i.e. *ground*). The other type of inferences add, as their fillers, parts of other role fillers in the conceptualization. For example, this sentence:

Vance left for Italy.

should build the following conceptualization:

(ACTOR *VANCE*
<=> *PTRANS*
OBJECT *VANCE*
FROM *USA*
TO *ITALY*)

The problem is that the sentence does not mention the U.S. at all. Nonetheless, the FROM role ought to be filled with *USA*. This is done by the second type of inference rule. The inference rule relevant here is

Primitive Act: *PTRANS*

Desired Role: (FROM)

Inferable Filler: NATIONALITY-OF (ACTOR)

Tests: (ACTOR) filled with a *HUMAN*
(TO) filled with a *COUNTRY* not NATIONALITY-OF (ACTOR)

The inference rule needed here states that if a person is *PTRANS*ed to a country, assume he started out at his home country unless he is going to his home country. The certainty of this rule is 6; it can be overwritten relatively easily. A different more certain inference rule would have been used if the system had known that Vance was previously in England.

Thus, FRUMP's role inferencer provides a method of satisfying the needs of PREDICTOR even though the information is not explicitly present in the text. FRUMP has approximately 50 such inference rules. They vary

greatly in the generality. The inference "if a plane crashes into something, the something is probably the ground" is very specific in its applicability. Most of the rules are more general. For example, there is an inference rule that says "the FROM role of an ATRANS can be filled with the previous owner of the object." That is, when something changes ownership, the entity giving up ownership is the old owner. We expect that there will be many more inference rules added to the system.

5.5 The Selection Procedure

This section describes how the SUBSTANTIATOR decides which of its structure building techniques to use for a given prediction.

Use of the conceptualization inferencer is straightforward. Recall that the conceptualization inferencer makes script-related inferences. When the PREDICTOR satisfies a predicted event, the selection process notifies the conceptualization inferencer. The inferencer then examines that sketchy script event. As described previously, if the script indicates that it must lead to, or must have been preceded by another script event, that event is inferred.

Managing the two role-filling routines, the parser and the role inferencer, is a bit more involved. Recall that each parsing rule and each inference rule has both a certainty and a cost associated with it.

The selection mechanism's job is to fill the desired role as certainly as possible without exceeding a predetermined cost. The maximum cost permitted is a parameter in the selection routine. Any rule that exceeds the cost parameter is not used by SUBSTANTIATOR. Thus with a higher cost parameter, more parsing and inference rules are available to substantiate predictions. When these more costly rules are available, FRUMP achieves a deeper and more certain but slower understanding of the text. Conversely, when the maximum cost is low, FRUMP does very fast but superficial processing.

The selection of the "best" rule is reasonably efficient. Recall that the role inference rules are organized by the primitive act of the conceptualization and within primitive acts by the role they fill. Furthermore, the inferences that fill the same role for the same primitive act are in order of decreasing certainty. The selection mechanism can efficiently find the inference rules that might add a certain role to a certain

conceptualization. Then the most certain inference rule is the first on the list. There is a static list of the parsing rules in decreasing certainty as well.

The best role-adding rule is then selected by a simple sort-merge technique. The cost of the more certain rule at the front of both lists is computed. If it is above the threshold, that rule is discarded. If not, that rule is applied. If the rule succeeds, the selection routine passes this information along to the PREDICTOR. In this way all of the applicable rules will be tested. If none can fill the role with the desired filler, the PREDICTOR is informed that its prediction must be wrong. In either case, PREDICTOR can then revise its predictions as described in chapter 4. We can now understand the SUBSTANTIATOR's behavior in the example the last section "The jet crashed." The SUBSTANTIATOR called the text analyzer to fill OBJECT role because the parse rules seemed the most promising (they had the highest certainties). Parse rule 4, which would have filled the role with *JET* was not tried because its certainty is less than the certainty of the inference rule that was used to fill the OBJECT role. Had the inference rule failed to fill the role, parse rule 4 might well have been tried.

Norman and Bobrow [1975] have proposed a slightly different and more general kind of resource limited processing. They advocate a method of processing which makes an initial guess at the solution to a task as quickly as possible. Further processing is then devoted to refining this guess. In this way the best guess at the solution is always maintained. Processing continues until a given allotment of a resource is exhausted at which time the result is taken to be the current "best" solution.

This scheme is particularly well suited when a process is given a fixed amount of some resource. However, such is not the case for FRUMP. Instead, it is more desirable to be able to tell FRUMP to speed up or slow down without regard to the absolute amount of any processing resource used. FRUMP can then dynamically adjust its skimming rate to the rate at which the input text is being presented. When the input text is arriving faster than FRUMP is processing it, the cost threshold is lowered to speed up processing. When FRUMP is ahead of the input, it raises the threshold to obtain a more certain and complete understanding.

There has recently been some psychological research into human skimming under time constraints by Kintsch & Masson (personal communication). They identify macro- and micro-propositions in the text. Macro-propositions are propositions which are central to the story. Micro-propositions correspond to less central facts. Their

initial results indicate that constraining processing time degrades comprehension of both macro- and micro-propositions quite uniformly.

In their terminology, FRUMP's sketchy scripts are comprised almost entirely of macro-propositions (i.e. events which are central to the script situation). Thus, the initial findings have no direct relevance to FRUMP's processing. What would be relevant is a quantitative specification of how comprehension of macro-propositions alone is degraded with processing constraints.

5.6 An Example

The following annotated example of FRUMP processing an input sentence illustrates the SUBSTANTIATOR's text analyzer and role inferencer processing, and demonstrates the intimate interaction between the PREDICTOR and the SUBSTANTIATOR. The input text sentence is:

> UGANDA TOOK FORMAL CONTROL OF AN AMERICAN OIL REFINERY.

Before this sentence was seen, PREDICTOR had already identified a sketchy script which predicted several conceptualizations. Among them is the following:

```
(ACTOR    (*POLITY*)

<=>       (*ATRANS*)

MANNER    (*FORCED*)

OBJECT    (*CONT*)
    TYPE      (*ECONOMIC*)
    PART      (*SPEC-INDUSTRY*)

TO        (*POLITY*)

FROM      (*POLITY*))
```

This is a conceptual dependency representation for the event of one country taking economic control of a specific industry from another. It is this prediction that FRUMP must match with the input. On the left below is the computer output generated during processing. Each output is prefaced with which module it came from. On the right are explanatory comments on the processing that resulted in the output message. It should be noted in reference to word

numbers that the computer begins numbering the input words from zero.

Input:

UGANDA TODAY TOOK FORMAL CONTROL OF AN AMERICAN OIL REFINERY.

COMPUTER OUTPUT	COMMENTS
SUBSTANTIATOR: ((<=> (*ATRANS*) MANNER (*FORCED*))) BUILT FROM WORD# (2) WORD SENSE TAKE1 PARTIALLY MATCHES A PREDICTION.	SUBSTANTIATOR looks for any word sense that could build a structure that partially matches a predicted concept. It found the word "took" with a word sense TOOK1 which matches a prediction.
PREDICTOR: PREDICTING ROLE (ACTOR) WILL BE FILLED WITH AN ELEMENT FROM THE LIST (*POLITY*)	PREDICTOR examines the partial conceptualization and predicts that the ACTOR role must be filled with a *POLITY*. The ACTOR of each prediction that could possibly be matched is filled with *POLITY*. Thus if this structure is going to match one of them, it must also have a *POLITY* ACTOR.
SUBSTANTIATOR: PREDICTING (ACTOR) IS SUBJECT OF (TAKE1 2 NIL PAST)	Using its syntactic knowledge SUBSTANTIATOR determines that the ACTOR will probably be found as the subject of the verb "took" from parse rule 1
FOUND POSSIBLE (*POLITY*) FROM WORD# (0) UGANDA (ACTOR) HAS BEEN FILLED WITH (*UGANDA*) (TO) HAS BEEN FILLED WITH (*UGANDA*)	Indeed, a *POLITY* was found where the syntactic subject was expected. Therefore it must be the conceptual ACTOR. The TO role is also filled with the same *POLITY* because the verb sense TAKE1 contains the information that its ACTOR and TO role fillers are the same.

PREDICTOR: PREDICTING ROLE (OBJECT) WILL BE FILLED WITH AN ELEMENT FROM THE LIST (*POSS* *CONT*)	There are several predicted conceptualizations that the partial conceptualization under construction can match. Some of them are abstract transfers of POSSession, others of CONTrol. Thus, to differentiate which prediction the text might satisfy, PREDICTOR asks that the OBJECT be filled with either *POSS* or *CONT*.
SUBSTANTIATOR: PREDICTING (OBJECT) IS VERB-OBJECT OF (TAKE1 2 NIL PAST)	SUBSTANTIATOR has used its syntactic knowledge to decide that if the conceptual OBJECT is specified in the text it will be the object of the verb "took" - parse rule 1 again.
FOUND POSSIBLE (*ABSTRACT*) FROM WORD# (4) (OBJECT) HAS BEEN FILLED (*CONT*)	At word number 4, SUBSTANTIATOR found what it was looking for: a word that means *CONT*.
PREDICTOR: PREDICTING ROLE (OBJECT PART) WILL BE FILLED WITH AN ELEMENT FROM THE LIST (*HUMAN* *SPEC-INDUSTRY*)	Again PREDICTOR is trying to differentiate between several viable predictions. The (OBJECT PART) must be filled with either a human or a specific industry.
SUBSTANTIATOR: WORD# (5) OF CAN POSSIBLY ADD (OBJECT PART) TENTATIVELY RESOLVING OF TO OF1	SUBSTANTIATOR found a preposition which it thinks can provide the desired information. This is from parse rule 3.
PREDICTING (OBJECT PART) IS PREP-OBJECT OF (OF1 5)	Here it is looking for the object of the preposition "of" at word 5.
FOUND POSSIBLE (*INDUSTRY*) FROM WORD# (9) (OBJECT PART) HAS BEEN FILLED WITH (*REFINERY*)	As the object of the preposition SUBSTANTIATE found "refinery" which it knows is a kind of industry.

PREDICTING (OBJECT PART CLASS) IS MODIFIER OF (REFINERY1 9) FOUND POSSIBLE (*PRODUCT*) FROM WORD# (8) (OBJECT PART CLASS) HAS BEEN FILLED WITH (*OIL*)	To be a specific industry, the kind of refinery must be determined. It decides that the kind of refinery, if present, will probably be an adjective modifier of "refinery" at word 9. It finds "OIL" at word 8.
PREDICTOR: PREDICTING ROLE (OBJECT TYPE) WILL BE FILLED WITH AN ELEMENT FROM THE LIST (*ECONOMIC*)	The PREDICTOR has by now narrowed down the number of viable predicted conceptualizations to one. That one requires that the type of control taken over the industry be economic.
SUBSTANTIATOR: PREDICTING (OBJECT TYPE) IS MODIFIER LOOKING FOR MODIFIER OF (CONT1 4 10) TEXT ANALYZER UNABLE TO FIND MODIFIER	SUBSTANTIATOR decides that if the OBJECT TYPE role is present in the text it will be as an adjective modifier of word 4 "control" as in "took economic control." However,the input phrase does not say "economic control" so the text analyzer cannot add the OBJECT TYPE role.
TRYING INFERENCE RULE INFERENCER ASKS SLOT (OBJECT BE FILLED WITH (*CONT*) (OBJECT) ALREADY FILLED WITH (*CONT*) INFERENCER ASKS SLOT (OBJECT PART) BE FILLED WITH *INDUSTRY* (OBJECT PART) ALREADY FILLED WITH (*REFINERY*) ALL TESTS FOR INFERENCE ARE TRUE---INFERRING (OBJECT TYPE) IS (*ECONOMIC*) (OBJECT TYPE) HAS BEEN FILLED WITH (*ECONOMIC* CERTAINTY (7))	SUBSTANTIATE decides to try to infer the desired role filler. It finds an inference rule that can add *ECONOMIC* in the OBJECT TYPE role of *ATRANS* acts provided certain conditions are met. Inference rules are indexed by the conceptual act, and the role they add. Thus they can be found efficiently. The conditions required by this rule include that control of an industry be changing hands. If that is true, then the control is probably of type *ECONOMIC*.
PREDICTOR: PREDICTING ROLE (FROM) WILL	Finally PREDICTOR requests

```
    BE FILLED WITH AN ELEMENT  | that the FROM role be filled
    FROM LIST (*POLITY*)       | with a *POLITY*.
                               |
SUBSTANTIATOR:                 |
  TEXT ANALYZER UNABLE TO ADD  | However, SUBSTANTIATOR cannot
    (FROM) - CALLING           | add the FROM role using the
    INFERENCE PROCEDURES       | text.
                               |
  TRYING INFERENCE RULE        | An inference rule is found
  INFERENCER ASKS SLOT         | which says that for abstract
    (OBJECT PART OWNER) BE     | transfers the entity giving
    FILLED WITH *POLITY*       | up the object is probably the
                               | same as the current owner of
                               | the object. Thus the problem
                               | has been reduced to finding
                               | the OBJECT PART OWNER.
                               |
  FILLER MISSING -             |
    SUBSTANTIATOR CALLED       |
  PREDICTING (OBJECT PART      | SUBSTANTIATOR has decided
    OWNER) IS MODIFIER OF      | that if the owner is
    WORD# (9)                  | specified in the text it is
  LOOKING FOR MODIFIER OF      | probably an adjective
    (REFINERY1 9)              | modifier of "refinery" at
  FOUND POSSIBLE (*ANIMATE*)   | word 9. And indeed the owner
    FROM WORD# (7)             | is found to be the US.
  (OBJECT PART OWNER) HAS      |
    BEEN FILLED WITH (*USA*)   |
                               |
  ALL TESTS FOR INFERENCE ARE  | The inference is made.
    TRUE---INFERRING (FROM)    |
    IS (*USA* CERTAINTY (9))   |
                               |
PREDICTOR:                     |
  PREDICTED CONCEPTUALIZATION  | And finally the predicted
  SATISFIED:                   | conceptualization has been
  ((<=>      (*ATRANS*)        | fleshed out.
    MANNER   (*FORCED*))
    ACTOR    (*UGANDA*)
    TO       (*UGANDA*)
    OBJECT   (*CONT*
        TYPE     (*ECONOMIC* CERTAINTY (7))
        PART     (*REFINERY*)
            TYPE     (*OIL*)
            OWNER    (*USA*)
    FROM     (*USA* CERTAINTY (9)) ))
```

The conceptualization produced contains the information that the industry changing hands is an oil refinery of the United States, that the country taking it is Uganda, and that the country giving it up is the United States. All of

this was built in a very purposeful manner. The text was never examined without knowing what conceptual structure was to be built and approximately where in the text it would be found.

It seems as though a lot of work has been done to arrive at the correct parse of the sentence. Indeed, PREDICTOR and SUBSTANTIATOR each had to produce a large number of sub-results. However, each of these sub-results was achieved very efficiently. Very little work had to be done for any of them. The overall process is made much easier and more efficient from the exchange of information between PREDICTOR and SUBSTANTIATOR.

CHAPTER 6

PREDICTOR/SUBSTANTIATOR INTERACTION

6.1 Introduction

Now that the operations of the PREDICTOR and SUBSTANTIATOR have been discussed in detail, we will examine their interaction during FRUMP's processing of a news story. The system focuses its attention on building one conceptual structure at a time. In general, the PREDICTOR asks the SUBSTANTIATOR to add some piece of conceptual structure to the existing conceptualization. The actual prediction is based on what has been built up previously in processing the conceptualization and other conceptualizations from the input. The SUBSTANTIATOR then tries to add something to the current conceptualization that will satisfy the PREDICTOR. Based on the actions of the SUBSTANTIATOR, the PREDICTOR reassesses the context and makes another prediction. In this way, the text is always interpreted against a background of expectations supplied from knowledge of the world.

We will discuss the processing of the following article:

> New York, May 30 -- A federal grand jury yesterday charged Luis Serez with first degree murder in connection with the May 16 slaying of a New York City police officer. Officer Fredrick Govel was allegedly shot by Serez when he attempted to quell a barroom dispute in a lower Manhattan tavern. After the shooting Serez escaped through an alley service entrance.

> Police apprehended Serez last Thursday when he succumbed to a police ruse and ran out of his West 19th Street apartment building into the hands of waiting detectives. A police detective reportedly phoned Serez saying "Some guy said call this number and say the police are on the way to get you," and then hung up. Within moments Serez came running out of the building.

This article is largely from a front page New York Times news story. However, it has been shortened considerably and altered somewhat in order to illustrate several additional processing points. The original story reported only the arrest, not an indictment.

FRUMP decided that the story described facts preliminary to a trial. It therefore used the sketchy script $COURT by event induced activation. In processing the story, the sketchy scripts $MURDER and $ARREST were also instantiated. These scripts were activated by implicit reference via an issue skeleton as discussed in chapter 3. Issue skeletons will be denoted by a word in capital letters preceded by "%". The issue skeleton relevant to this story is %CRIME.

6.2 The Sketchy Scripts Involved

The scripts involved in understanding this story are $COURT, $MURDER, and $ARREST. Before observing FRUMP processing the story, we will briefly examine the relevant conceptualizations from the three sketchy scripts $CRIME, $MURDER, and $ARREST.

The first sketchy script is $COURT. The conceptualization that will be matched in this script represents the indictment. The corresponding conceptual dependency representation is:

```
(ACTOR        *GRAND-JURY*
 <=>          *MTRANS*
 MOBJECT      *ACCUSATION*
   SUSPECT      *HUMAN*
   CRIME        *CRIME*
 FROM         *GRAND-JURY*
 TO           *HUMAN*)
```

This says that a grand jury is communicating charges to the suspect of a crime. "Indictment" is only the first conceptualization of $COURT. The sketchy script has many other conceptualizations in it as well. However, this is the only representation that is relevant to the example story.

It should be noted that the *HUMAN* in the MOBJECT SUSPECT role and the *HUMAN* in the TO role must be the same. This is not explicitly indicated in the above representation, but the constraint is present in the sketchy script from the fact that they are bound to the same script variable.

The next relevant sketchy script is $MURDER. It contains conceptualizations that represent the usual ways to murder someone, and whether or not the victim died. There are two conceptualizations important for the example story. They are:

```
(ACTOR          *HUMAN*

 <=>            *PROPEL*

 OBJECT         *BULLET*

 FROM           *GUN*

 TO             *HUMAN*)
```

This is the conceptual representation for the action of shooting at someone. The next conceptualization represents the fact that someone has died:

```
(ACTOR          *HUMAN*

 IS             *HEALTH*
   VAL            -10   )
```

In addition to the representations, there exists knowledge about which roles must be coreferent (the person dying must be the person shot). There is also knowledge about causal connections between conceptualizations (the person died because he was shot).

The sketchy script $MURDER is used for processing attempted murders as well as successful ones. Thus this predicted event need not always be found or inferred in a story. In processing the example story, FRUMP satisfied both of the above events.

The next two conceptualizations represent the relevant facts in the sketchy script $ARREST:

```
(ACTOR          *POLICE*

 <=>            *ATRANS*

 OBJECT         *CONT*
   TYPE           *SOCIAL*
   PART           *HUMAN*

 FROM           *HUMAN*

 TO             *POLICE*)
```

This is the conceptual representation of arresting a person. An arrest is an abstract transfer of social control of a person from himself to the police. There is an additional constraint that the OBJECT PART filler and the filler of the FROM role be the same *HUMAN*. Again this is supplied by the fact that the two fillers must be bound to the same script variable. Finally in the arrest sketchy script, the police might charge the suspect. The representation of this is the following:

```
(ACTOR          *POLICE*

 <=>            *MTRANS*

 MOBJECT        *ACCUSATION*
   SUSPECT        *HUMAN*
   CRIME          *CRIME*

 FROM           *POLICE*

 TO             *HUMAN*)
```

The representation of "charging" is very similar to the representation for "indicting" that was used in $COURT. The difference is that to be an indictment, the charge must be made by a grand jury, while in the arrest script, police bring the charges.

Now that we have examined the relevant conceptual representations, we are ready to consider FRUMP's processing. The remainder of the chapter will be devoted to explaining how FRUMP finds instances of these conceptual representations in the example story.

6.3 An Annotated FRUMP Run

During our discussion of FRUMP's processing of this story we will use the convention of placing the computer input and output in capital letters and the comments in lower case. Each of the FRUMP processing messages is prefaced by the module that generated it.

INPUT:

NEW YORK, MAY 30 -- A FEDERAL GRAND JURY YESTERDAY CHARGED LUIS SEREZ WITH FIRST DEGREE MURDER IN CONNECTION WITH THE MAY 16 SLAYING OF A NEW YORK CITY POLICE OFFICER. OFFICER FREDRICK GOVEL WAS ALLEGEDLY SHOT BY SEREZ WHEN HE ATTEMPTED TO QUELL A BARROOM DISPUTE IN A LOWER MANHATTAN TAVERN. AFTER THE SHOOTING SEREZ ESCAPED THROUGH AN ALLEY SERVICE ENTRANCE.

POLICE APPREHENDED SEREZ LAST THURSDAY WHEN HE SUCCUMBED TO A POLICE RUSE AND RAN OUT OF HIS WEST 19TH STREET APARTMENT BUILDING INTO THE HANDS OF WAITING DETECTIVES. A POLICE DETECTIVE REPORTEDLY PHONED SEREZ SAYING "SOME GUY SAID CALL THIS NUMBER AND SAY THE POLICE ARE ON THE WAY TO GET YOU." AND THEN HUNG UP. WITHIN MOMENTS SEREZ CAME RUNNING OUT OF THE BUILDING.

SUBSTANTIATOR:
SEARCHING FOR A STRUCTURE BUILDING WORD

((<=> (*MTRANS*) MOBJECT (*ACCUSATION*))) BUILT FROM WORD# (15) CHARGED

PREDICTOR:
PREDICTING ROLE (<=>) WILL BE FILLED WITH AN ELEMENT FROM THE LIST (*ATRANS* *PROPEL* *PTRANS* *MTRANS*)

(<=>) ALREADY FILLED WITH *MTRANS*

At this point FRUMP is trying to figure out what sketchy script, if any, is appropriate to use in understanding the story. To start things off, the SUBSTANTIATOR asks the text analyzer to build anything it can.

The text analyzer finds the first word that has a Structure Building Word Sense. This word is "charge" which has a word sense CHARGE1 that builds an *MTRANS* of an *ACCUSATION*. It was accidental that the correct word sense

of "charge" was chosen first. FRUMP might as easily have resolved "charge" to "demand payment" or "violently attack." However, these would have been rejected very quickly by later predictions. Had it resolved the word in a different way no later predictions would be satisfiable from the input.

The word "charged" is the fifteenth word. This is stored by the text analyzer for future reference. In FRUMP's word numbering scheme, punctuation and capitalization are represented explicitly in the text and therefore are numbered as well.

Once some structure has been built, the PREDICTOR asks that the action role (in CD the <=> role) be filled and predicts that it will be filled with one of a list of primitive acts. This step is necessary for the PREDICTOR to begin traversing the Sketchy Script Indicator Discrimination Tree described in chapter 3. The primitive acts in the list are the combined different acts that appear in key requests of all of the sketchy scripts in the system. Recall that a key request is a conceptualization that indicates a particular sketchy script situation is being described.

The predicted role already has been filled with one of the desired items (*MTRANS*).

```
PREDICTOR:
  PREDICTING ROLE (ACTOR) WILL BE FILLED WITH AN
    ELEMENT FROM THE LIST (*HUMAN* *POLITY* *POLICE*
    *GRAND-JURY* *ANIMATE* *AUTHORITIES*)


SUBSTANTIATOR:
  PREDICTING (ACTOR) IS SUBJECT OF (CHARGE1 15 NIL PAST)

  FOUND POSSIBLE (*GRAND-JURY*) FROM WORD# (13) JURY
```

The PREDICTOR follows the *MTRANS* arc to the next node in the sketchy script discrimination tree. The next node tests the ACTOR role. Again there is a list of possible fillers, any one of which will permit the structure being built as a viable key request.

The SUBSTANTIATOR selects parse rule 1 because there is a word that has been processed which can predict the syntactic location of the conceptual ACTOR filler. CHARGE1 predicts that its syntactic subject will be the conceptual ACTOR. The text analyzer scans for the syntactic subject and finds the word "jury." The lexical item "jury" has several word senses one of which is a phrase word sense. Phrase word senses were discussed in chapter 5. The phrase

of this word sense requires the word "jury" to be preceded by the word "grand." If this is the case, it can resolve to a word sense, GRANDJURY1, which has as its conceptual entry *GRAND-JURY*. Since "grand" does precede "jury," and *GRAND-JURY* matches at least one of the predicted items, "jury" is resolved to GRANDJURY1.

```
(SUBSTANTIATOR:)
  (ACTOR) HAS BEEN FILLED WITH (*GRAND-JURY*)
  (FROM) HAS BEEN FILLED WITH (*GRAND-JURY*)
```

The word sense CHARGE1 contains the information that the entity doing the charging is both the filler of the ACTOR role and the filler of the FROM role in the underlying conceptualization. Therefore, they are both filled.

```
PREDICTOR:
  PREDICTING ROLE (MOBJECT) WILL BE FILLED WITH AN
    ELEMENT FROM LIST (*ACCUSATION*)
  (MOBJECT) ALREADY FILLED WITH (*ACCUSATION*)
```

The PREDICTOR now asks that the MOBJECT be filled with something that can be considered an *ACCUSATION*. However, the MOBJECT role is already filled with *ACCUSATION* from the structure built by CHARGE1. Therefore, the SUBSTANTIATOR need not be called.

At this point we will make a slight digression to discuss a problem that might occur. The SUBSTANTIATOR filled the ACTOR role with *GRAND-JURY* on the assumption that the conceptual referent for the word "jury" should be interpreted as *GRAND-JURY*. It need not be interpreted that way. The *GRAND-JURY* built from the word "jury" can match several items in the list of predicted fillers. It can match *HUMAN*, *GRAND-JURY*, and *ANIMATE*. Thus the filler of the ACTOR role can be considered a *HUMAN* or an *ANIMATE* as well. However, the SUBSTANTIATOR used the tightest match. *GRAND-JURY* built by the text analyzer can indeed be considered *ANIMATE*, but several levels of hierarchical inheritance must be followed to match it. On the other hand, the predicted conceptual entity *GRAND-JURY* matches immediately. The SUBSTANTIATOR always tries the closest match first. The interpretation can, however, later be altered. If, for example, the correct interpretation were *ANIMATE* the SUBSTANTIATOR could reinterpret elements of the conceptualization built. Thus the system can recover from errors of this kind.

After noticing the MOBJECT is filled with *ACCUSATION*, FRUMP has narrowed the number of viable key requests to two, both of which are in the sketchy script $COURT. The two

possible matches are representations of the grand jury indicting a person and the grand jury acquitting a person.

```
PREDICTOR:
  PREDICTING ROLE (MODE) WILL BE FILLED WITH AN
    ELEMENT FROM THE LIST (*POS* *NEG*)


SUBSTANTIATOR:
  TEXT ANALYZER UNABLE TO ADD (MODE)  - CALLING
    INFERENCE PROCEDURES

  TRYING INFERENCE RULE
  ALL TESTS FOR INFERENCE ARE TRUE -- INFERRING
    (MODE) IS (*POS* CERTAINTY (8))
```

With this prediction, PREDICTOR is trying to differentiate between the two viable key conceptualizations corresponding to a grand jury indicting a person and failing to indict him. The difference is that the MODE role is filled with *POS* in one and *NEG* in the other. No parsing rule was able to add the desired role filler so the role inferencer was called. There is an inference rule that permits filling the MODE role with *POS*. The certainty of this rule is 8. Since it is less certain than the parsing rules, it will not be used unless the parsing rules fail. In effect, the rule says that if the text analyzer cannot justify filling the MODE role with anything else, assume that it is filled with *POS*.

```
PREDICTOR:
  PREDICTING ROLE (MOBJECT SUSPECT) WILL BE FILLED
    WITH AN ELEMENT FROM THE LIST (*HUMAN*)
```

Finally, the PREDICTOR requests that the MOBJECT SUSPECT role be filled with a *HUMAN*. This is the final role needed to match the conceptualization built from the input to a key conceptualization of the $COURT sketchy script.

```
SUBSTANTIATOR:
  PREDICTING (MOBJECT SUSPECT) IS THE VERB-OBJECT
    OF (CHARGE1 15 NIL PAST)

  FOUND POSSIBLE (*NAME*) FROM WORD# (19) SEREZ
  ADDING *NAME* WORD SENSE TO LEXICAL ITEM SEREZ
  (MOBJECT SUSPECT) HAS BEEN FILLED WITH (HUM1)
  (TO) HAS BEEN FILLED WITH (HUM1)
```

The SUBSTANTIATOR applies parse rule 1 which predicts the MOBJECT SUSPECT will be found as the syntactic VERB-OBJECT. In the position of the syntactic object of the verb, the text analyzer finds what can be interpreted as a person's name. The text analyzer has heuristics about what a name looks like. For example, it must be capitalized, it may begin with a title or a known first name, etc. The text analyzer also knows that when looking for a *HUMAN* it is sufficient to find a name. Thus "Luis Serez" is interpreted as the name of the *HUMAN* being looked for. At this point, a word sense, HUM1, is created and added as a word sense of the last name found, "Serez." This word sense has a conceptual entry which is a *HUMAN* and has the first name of LUIS and the last name of SEREZ.

```
PREDICTOR:
  SELECTED SKETCHY SCRIPT $COURT VIA REQUEST R1

  RELEVANT ISSUE SKELETONS ARE (%CRIME)
  SCRIPT CANNOT BE INCORPORATED INTO PREVIOUS ISSUE
    SKELETONS

  CREATING NEW SKETCHY SCRIPT SC0
  CREATING NEW ISSUE SKELETON IS0
```

The PREDICTOR has now identified the sketchy script $COURT. It also looks for any existing %CRIME issue skeleton. Since none is found, it creates a new one.

At this point the conceptualization built is:

```
(ACTOR          *GRAND-JURY*

 <=>            *MTRANS*

 MOBJECT        *ACCUSATION*
   SUSPECT        HUM1

 FROM           *GRAND-JURY*

 TO             HUM1

 MODE           *POS*
```

This is the minimum conceptual structure needed to index the $COURT sketchy script. It must be specified by a person when the script is written. The court sketchy script can appear in two issue skeletons: %CRIME and %CIVIL-SUIT. However, the conceptualization just built is incompatible with %CIVIL-SUIT. There is no indictment in a civil suit. However, it fits well with %CRIME. Therefore, the issue skeleton %CRIME is selected. The PREDICTOR looks for

previous %CRIME issue skeletons which this story might update. When it finds none, it creates new instances of the %CRIME issue skeleton and $COURT sketchy script.

The %CRIME issue skeleton looks like this:

THE CRIME ISSUE SKELETON

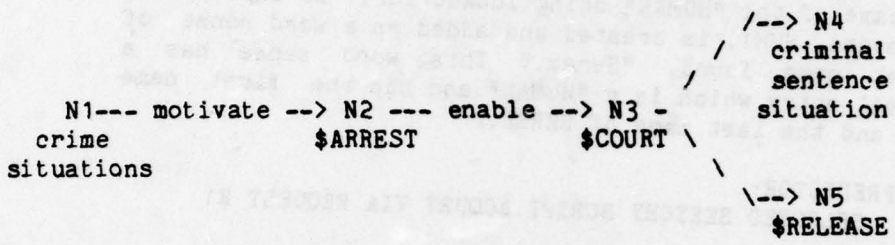


figure 6.1

Recall from chapter 3 that sketchy scripts related to the issue skeleton are activated when it is initiated. This is activation by implicit reference. Therefore, the sketchy scripts $ARREST and $RELEASE will be activated in addition to $COURT. Furthermore, there is an empty slot for the crime situation. The PREDICTOR will try to fit future instantiated sketchy scripts that can be considered "crime situations" (such as $KIDNAP and $MURDER) into this issue skeleton. The same will be done for "criminal sentence situations." This is the way FRUMP identifies the expected, but missing, crime.

Before FRUMP tries to find instances of the new predicted events, it tries to fill out the current *indictment* prediction as much as possible. It continues to predict missing roles in the conceptualization. The difference is that now the predictions are based on what was found in the $COURT sketchy script.

```
PREDICTOR:
  PREDICTING ROLE (TIME) WILL BE FILLED WITH AN
    ELEMENT FROM THE LIST (*TIME*)

SUBSTANTIATOR:
  FOUND POSSIBLE (*TIME*) FROM WORD# (14) YESTERDAY
  (TIME) HAS BEEN FILLED WITH (TIM1)
```

The text analyzer found the word "yesterday" and created a conceptual item TIM1 for it. The only word sense of "yesterday" builds a time which is one day before the current date. The current date is initially set to the dateline date. Thus TIM1 refers to May 29 and the current year.

```
SUBSTANTIATOR:
  PREDICTING (MOBJECT CRIME) IS WITH PREP-OBJECT OF
    (CHARGE1 15 NIL PAST)

  FOUND POSSIBLE (*CRIME*) FROM WORD# (23) MURDER

  (MOBJECT) HAS BEEN FILLED WITH ($MURDER)
```

The SUBSTANTIATOR decides to use the text analyzer to fill the MOBJECT role. The text analyzer uses parse rule 1 to predict that filler of the MOBJECT role will be found as the object of the preposition "with." The text analyzer looks where it expects the object of the preposition "with" and finds the word "murder." The word "murder" has a word sense that resolves to a reference to the sketchy script $MURDER. The sketchy script $MURDER is labeled with the fact that it can be considered a *CRIME*.

Thus the script at node N1 of the $CRIME issue skeleton is constrained to be the sketchy script $MURDER. Therefore, by implicit reference, that sketchy script is activated. Now the PREDICTOR is looking for the events in the sketchy scripts $MURDER, $ARREST, and $COURT. In addition, FRUMP will attempt to hook any criminal sentence sketchy script into this issue skeleton as well.

Also the indictment conceptualization in $COURT is complete. It looks like this:

```
(ACTOR          *GRAND-JURY*

 <=>            *MTRANS*

 MOBJECT        *ACCUSATION*
   SUSPECT        HUM1
   CRIME          $MURDER

 FROM           *GRAND-JURY*

 TO             HUM1)
```

PREDICTOR will ask SUBSTANTIATOR to build any conceptual structure that might match one of the predictions.

```
SUBSTANTIATOR:
  SEARCHING FOR A STRUCTURE BUILDING WORD

  ((TOWARD (*HEALTH* VAL (-10)))) BUILT FROM WORD#
    (31) SLAYING

PREDICTOR:
  PREDICTING ROLE (TOWARD) WILL BE FILLED WITH AN
    ELEMENT FROM THE LIST (*PHYSTATE* *HEALTH*)
  (TOWARD) ALREADY FILLED WITH *HEALTH*
```

The SUBSTANTIATOR finds the lexical item "slaying" which can build a state change. The PREDICTOR is only interested in certain state changes. One state change that is likely, given the current context, is a change in health. The PREDICTOR then notices that *HEALTH* already fills the TOWARD role.

```
PREDICTOR:
  PREDICTING ROLE (ACTOR) WILL BE FILLED WITH AN
    ELEMENT FROM THE LIST (*HUMAN*)

SUBSTANTIATOR:
  PREDICTING (ACTOR) IS VERB-OBJECT OF (SLAY1 31 GER NIL)

  FOUND POSSIBLE (*HUMAN*) FROM WORD# (40) OFFICER
  (ACTOR) HAS BEEN FILLED WITH (HUM2)
```

The PREDICTOR can only use this state change if the ACTOR is filled with a *HUMAN*. Therefore, it predicts that the ACTOR role will be filled with a *HUMAN*. The SUBSTANTIATOR predicts that the desired item will be found as the VERB-OBJECT of "slaying." Since "slaying" is a gerund, syntax rule 9 dictates that the desired item will probably be the object of the preposition "of." The SUBSTANTIATOR finds the word "officer" which has a phrase word sense "police officer." The phrase word sense of "officer" is similar to the phrase word sense used for "grand jury." This phrase word sense of "officer" resolves to *POLICEMAN* which is an occupation. The text analyzer knows that occupations are often used to refer to a person with that occupation. A new conceptual entity HUM2 is created. It has *HUMAN* as its conceptual referent and also contains the fact that the occupation of the *HUMAN* is a policeman.

```
PREDICTOR:
  PREDICTING ROLE (TOWARD VAL) WILL BE FILLED WITH AN
    ELEMENT FROM THE LIST (*LTO* -10)

  (TOWARD VAL) ALREADY FILLED WITH (-10)
```

Finally, the PREDICTOR must differentiate between the $MURDER predictions of someone dying and just being injured. In an attempted murder, for example, the victim does not die but might be injured. However, the (TOWARD VAL) role is already filled with -10.

The conceptualization built now matches a predicted conceptualization, that of a person dying. Just as before there are optional roles. These need not be filled in order to match the predicted conceptualization. They do, however, represent information that is important to find from the text. The PREDICTOR now tries to fill these optional role.

```
PREDICTOR:
  PREDICTING ROLE (TIME) WILL BE FILLED WITH AN
    ELEMENT FROM THE LIST (*TIME*)

SUBSTANTIATOR:
  FOUND POSSIBLE (*TIME*) FROM WORD# (29) MAY

  (TIME) HAS BEEN FILLED WITH (TIM2)
```

The optional role TIME was predicted. SUBSTANTIATOR found a possible time from the word "May." A new token TIM2 was created to represent the time of the event. TIM2 contains the date May 16, 1979. *TIME* is a composite role filler which has parts of a year, a month, a day, and a daytime. When such a composite is predicted, SUBSTANTIATOR tries to build all of its parts. In this example, the text analyzer was able to add the month and the day. The year was added by an inference rule that supplies the current year.

Now this predicted conceptualization is complete and FRUMP is ready to build another one.

```
SUBSTANTIATOR:
  SEARCHING FOR A STRUCTURE BUILDING WORD

  ((<=> (*PROPEL*) OBJECT (*BULLET*) FROM (*GUN*)))
    BUILT FROM WORD# (51) SHOT

PREDICTOR:
```

```
      PREDICTING ROLE (<=>) WILL BE FILLED WITH AN
        ELEMENT FROM THE LIST (*ATRANS* *PROPEL*
        *PTRANS* *MTRANS*)

      (<=>) ALREADY FILLED WITH *PROPEL*

      PREDICTING ROLE (ACTOR) WILL BE FILLED WITH AN
        ELEMENT FROM THE LIST (*HUMAN* *POLICE*)

    SUBSTANTIATOR:
      PREDICTING (ACTOR) IS SUBJECT OF (SHOOT1 51 PASS PAST)

      FOUND POSSIBLE (*HUMAN*) FROM WORD# (54) SEREZ
      (ACTOR) HAS BEEN FILLED WITH (HUM1)
```

Now the *PREDICTOR has built the conceptualization of* a *PROPEL* with the ACTOR filled by HUM1, "Serez." The text analyzer knows that "shot" is passive and so applies syntax rule 6. It therefore looks at the object of the preposition "by" instead of the syntactic subject for a *HUMAN*. HUM1 is found as a word sense of "Serez." Recall that when FRUMP first recognized the name, it created the conceptual token HUM1 for it which was then added to the word senses of "Serez."

```
    PREDICTOR:
      PREDICTING ROLE (OBJECT) WILL BE FILLED WITH AN
        ELEMENT FROM THE LIST (*PHYSOBJ* *KNIFE*
        *BULLET* *AXE*)
      (OBJECT) ALREADY FILLED WITH (*BULLET*)

      PREDICTING ROLE (FROM) WILL BE FILLED WITH AN
        ELEMENT FROM THE LIST (*GUN*)
      (FROM) ALREADY FILLED WITH (*GUN*)

      PREDICTING ROLE (TO) WILL BE FILLED WITH AN
        ELEMENT FROM THE LIST (HUM2)

    SUBSTANTIATOR:
      PREDICTING (TO) IS VERB-OBJECT OF
        (SHOOT1 51 PASS PAST)

      FOUND POSSIBLE (HUM2) FROM WORD# (48) GOVEL
      (TO) HAS BEEN FILLED WITH (HUM2)
```

The text analyzer finds a possible name with the title "officer" where it expects to find HUM2. Recall that HUM2 is a *HUMAN* with the occupation of *POLICEMAN*. Since the title officer does not conflict with this occupation, and

since HUM2 does not yet have an assigned name, the name found is merged into the created token HUM2.

At this point the PREDICTOR has built a conceptualization that matches the predicted event of a shooting as the cause of the death. Therefore, that event in the sketchy script $MURDER is satisfied. Finally the PREDICTOR tries to add the optional role TIME. However, this fails.

Now the PREDICTOR has finished its processing of this phrase. Once again it asks the SUBSTANTIATOR to build any partial conceptualization it can. The PREDICTOR will then try to fill out the partial conceptualization to satisfy another script conceptualization. The SUBSTANTIATOR looks for a structure building word. There are several structure words such as "quell," "dispute," etc. However, none of these are in phrases that can satisfy any script events. Each is rejected after minimal processing. That is, after it is given each of these partial conceptualizations, the PREDICTOR makes other predictions to augment them so as to match a script event. The SUBSTANTIATOR is unable to satisfy any of these predictions.

The next interesting structure building word that is found is "apprehended" in the second paragraph.

```
SUBSTANTIATOR:
  SEARCHING FOR STRUCTURE BUILDING WORDS
  BUILT ((<=> (*ATRANS*) OBJECT
    (*CONT* TYPE (*SOCIAL*)))) FROM WORD#
    (86) APPREHENDED


PREDICTOR:
  PREDICTING ROLE (<=>) WILL BE FILLED WITH AN
    ELEMENT FROM THE LIST (*ATRANS* *PROPEL*
    *PTRANS* *MTRANS*)
  (<=>) ALREADY FILLED WITH (*ATRANS*)

  PREDICTING ROLE (ACTOR) WILL BE FILLED WITH AN
    ELEMENT FROM THE LIST (*POLICE*)


SUBSTANTIATOR:
  PREDICTING (ACTOR) IS SUBJECT OF
    (APPREHEND1 86 NIL PAST)

  FOUND POSSIBLE (*POLICE*) FROM WORD# (85) POLICE
  (ACTOR) HAS BEEN FILLED WITH (*POLICE*)
  (TO) HAS BEEN FILLED WITH (*POLICE*)
```

```
PREDICTOR:
  PREDICTING ROLE (OBJECT) WILL BE FILLED WITH AN
    ELEMENT FROM THE LIST (*CONT*)
  (OBJECT) ALREADY FILLED WITH (*CONT*)

  PREDICTING ROLE (TO) WILL BE FILLED WITH AN
    ELEMENT FROM THE LIST (*POLICE*)
  (TO) ALREADY FILLED FILLED WITH (*POLICE*)

  PREDICTING ROLE (FROM) WILL BE FILLED WITH AN
    ELEMENT FROM THE LIST (HUM1)

SUBSTANTIATOR:
  PREDICTING (FROM) IS VERB-OBJECT OF
    (APPREHEND1 86 NIL PAST)

  FOUND POSSIBLE (HUM1) FROM WORD# (88) SEREZ
  (FROM) HAS BEEN FILLED WITH (HUM1)
  (OBJECT PART) HAS BEEN FILLED WITH (HUM1)


PREDICTOR:
  PREDICTING ROLE (OBJECT TYPE) WILL BE FILLED WITH
    AN ELEMENT FROM THE LIST (*SOCIAL*)
  (OBJECT TYPE) ALREADY FILLED WITH (*SOCIAL*)

  PREDICTING ROLE (TIME) WILL BE FILLED WITH AN
    ELEMENT FROM THE LIST (*TIME*)


SUBSTANTIATOR:
  FOUND POSSIBLE (*TIME*) FROM WORD# (91) THURSDAY
  (TIME) HAS BEEN FILLED WITH (TIM3)


PREDICTOR:
  SCRIPT EVENT SATISFIED


SUBSTANTIATOR:
  INFERRING SCRIPT EVENT ((ACTOR (*POLICE*) <=> (*MTRANS*)
    MOBJECT (*ACCUSATION* SUSPECT (HUM1) CRIME ($MURDER))
    FROM (*POLICE*) TO (HUM1)))
```

Here, FRUMP satisfies the conceptualization for the police arresting the suspect. A script inference from that event is that the police charge the suspect. The conceptualization inferencer of the SUBSTANTIATOR infers that event when the arrest event has been built.

The final representation built for this story consists of an instance of the %CRIME issue skeleton and three instantiated sketchy scripts, $MURDER, $ARREST, and $COURT:

CONCEPTUAL STRUCTURE BUILT

```
%CRIME001:                                              /--> N4
                                                      /    criminal
                                                    /      sentence
   N1-- motivate ---> N2 --- enable --> N3 -              situation
 $MURDER001          $ARREST001       $COURT001 \
                                                 \
                                                  \--> N5
                                                    $RELEASE
```

```
$MURDER001:
     (ACTOR          HUM1
      <=>            *PROPEL*
      OBJECT         *BULLET*
      FROM           *GUN*
      TO             HUM2)


     (ACTOR          HUM2
      IS             *HEALTH*
        VAL            -10   )
```

```
$ARREST001:
     (ACTOR          *POLICE*
      <=>            *ATRANS*
      OBJECT         *CONT*
        TYPE           *SOCIAL*
        PART           HUM1
      FROM           HUM1
      TO             *POLICE*
      TIME           TIM3)


     (ACTOR          *POLICE*
      <=>            *MTRANS*
      MOBJECT        *ACCUSATION*
        SUSPECT        HUM1
        CRIME          $MURDER
      FROM           *POLICE*
      TO             HUM1)
```

```
$COURT001:
     (ACTOR          *GRAND-JURY*
      <=>            *MTRANS*
      MOBJECT        *ACCUSATION*
        SUSPECT         HUM1
        CRIME           *CRIME*
      FROM           *GRAND-JURY*
      TO             HUM1
      MODE           *POS*
      TIME           TIM1)
```

ENGLISH SUMMARY:

LUIS SEREZ MURDERED FREDRIC GOVEL. POLICE ARRESTED HIM ON MAY 22, 1975 AND CHARGED HIM WITH MURDER. HE WAS INDICTED FOR MURDER ON MAY 29, 1975.

CHAPTER 7

ANNOTATED FRUMP OUTPUT

7.1 Introduction

This chapter contains fourteen different UPI stories and an explanation of FRUMP's processing of them. These are actual stories collected from the UPI wire between June 20, 1978 and March 15, 1979. With the exception of the second time FRUMP processed the fifth, eleventh, and thirteenth stories, neither the FRUMP program nor the stories were altered in any way.

Even though the texts are completely in upper case, FRUMP can tell which words are capitalized. The raw text input contains coded information that enables FRUMP to determine capitalized words.

Only English is produced by the generator FRUMP used in processing these stories. There are other generators with which FRUMP can be run that generate Spanish, Chinese, and French. Some examples of these were shown in chapter 1. However, those generators are less robust and consequently have very little chance of producing correct natural language outputs from novel inputs.

These stories were chosen to illustrate the processing that FRUMP does and some problems that arise when FRUMP deals with new input.

Just before the summary of each story FRUMP prints the processing time taken for that text. FRUMP runs on a DEC PDP 20/50 with 256K words of real memory.

7.2 The Stories

UPI Story 1, December 5, 1978
INPUT:

CAIRO, EGYPT (UPI)-EGYPT TODAY ANNOUNCED IT WAS BREAKING OFF DIPLOMATIC RELATIONS WITH BULGARIA.

THE MOVE WAS ANNOUNCED IN A FOREIGN MINISTRY STATEMENT. IT CAME AFTER BULGARIA RETALIATED AGAINST A POLICE RAID ON ITS EMBASSY IN CAIRO BY ORDERING THE EGYPTIAN AMBASSADOR AND HIS STAFF TO LEAVE WITHIN THREE DAYS AND RECALLING ITS OWN DIPLOMATIC LEGATION.

```
SELECTED SKETCHY SCRIPT $BREAK-RELATIONS

      BINDINGS          (&SIDE1 SV155 &SIDE2 SV156 &LEVEL
 SV157)
      REQUESTS          (R1 RQ80 R2 RQ81)
      INSTANCE          $BREAK-RELATIONS
REQUESTS:
 RQ80
  SATISFIED = T
     ((ACTOR &SIDE1 IS (*LINK* TYPE (*DIPLOMATIC*)
       LEVEL &LEVEL) WITH &SIDE2) MODE (*TF*))
 RQ81
  SATISFIED = INFERRED
     ((ACTOR &SIDE2 IS (*LINK* TYPE (*DIPLOMATIC*)
       LEVEL &LEVEL) WITH &SIDE1) MODE (*TF*))

 SV155
      WORD#     (68)
      CERTAINTY         (10)
      TYPE      (*POLITY*)
      CONENT    (*EGYPT*)
 SV156
      WORD#     (79)
      CERTAINTY         (10)
      TYPE      (*POLITY*)
      CONENT    (*BULGARIA*)
 SV157
      WORD#     (0)
      CERTAINTY         (4)
      TYPE      (*DPL-LEVEL*)
      CONENT    (*AMBASSADORIAL*)

CPU TIME FOR UNDERSTANDING = 3531 MILLISECONDS

ENGLISH SUMMARY:
      THE ARAB REPUBLIC OF EGYPT ENDED DIPLOMATIC RELATIONS
```

WITH BULGARIA.

When FRUMP is finished, it prints out the instantiated sketchy script as a closure. To find the filler of the conceptual role in one of the conceptualizations, one must find the script variable that fills the role in the list of BINDINGS. The GENSYMed atom following the script variable in the BINDINGS list is the memory node created for the filler. These nodes are printed along with their properties after the conceptualizations. For example, the filler of the ACTOR role in the first conceptualization is the script variable &SIDE1. The identifier that follows &SIDE1 in the BINDINGS list is SV155. The properties of SV155 are that it comes from word 68, "Egypt," the script variable is bound with a certainty of 10 (very certain), it is a type of POLITY, and it resolves to the permanent memory token *EGYPT*, where FRUMP's static knowledge about that country is stored.

The word numbering seems to be off because each UPI story is preceded by a header, composed largely of special symbols. Each header contains the date and time of transmission of the story, the reporting point, the reporter's name, a story identifier, etc. This is totally ignored by FRUMP except for the the headline information (the reporting point and the date).

This story states that Egypt ended diplomatic relations with Bulgaria. FRUMP represents one country having diplomatic relations with another as a state existing between them (a DIPLOMATIC LINK from one to the other). The breaking of relations is represented as the termination of the state. This is represented by MODE (*TF*), the Time of Finishing of the state.

FRUMP built a conceptual representation of Egypt ending relations with Bulgaria. It also inferred, via its conceptualization inferencer, that Bulgaria also ended diplomatic relations with Egypt. The CD Role Inferencer of the SUBSTANTIATOR supplied the inference that the DIPLOMATIC LINK was at the AMBASSADORIAL level. The Role Inferencer contains a rule that the default level of diplomatic relations is ambassadorial. That is, if no level could be found (e.g. consular) then assume ambassadorial level. Notice that the certainty of this script variable binding is only 4 while the others are 10.

FRUMP selected the sketchy script $BREAK-RELATIONS by Event Induced Activation. The script was selected on the basis of a conceptualization built. There is no single word contained in this article that can legitimately be marked as referring to the script for breaking relations. Neither the

word "broke" nor the phrase "broke off" should have a word sense specific to countries breaking relations. There are simply too many different situations at this level of specificity where "broke" can be used. The same is true for "relations," and, to a lesser extent, for "diplomatic."

Nor should the entire phrase "broke off diplomatic relations" be tagged with $BREAK-RELATIONS. There are too many such phrases. The article might have stated the event as "broke formal relations," "ended diplomatic ties," "terminated formal relations," etc. It is infeasible to anticipate them all.

Instead, FRUMP uses more general definitions of the words (e.g. "broke" has a sense that means "ending a state"). As explained in chapter 3 an entire conceptualization built from the text is used to activate a sketchy script. This script is activated by a conceptualization representing the time of ending of a DIPLOMATIC LINK state between two countries.

It should be pointed out that FRUMP did not resolve the pronoun "it." The version of the program that processed this story did not treat anaphora. Rather it found the word "Egypt" before the verb "break" where it expected the subject to be.

UPI Story 2, February 5, 1979
INPUT:

NAIROBI, KENYA (UPI)-UGANDAN AUTHORITIES HAVE ARRESTED SABOTEURS SENT IN FROM TANZANIA WHO EXPLODED BOMBS THAT BLACKED OUT MANY PARTS OF THE CAPITAL OF KAMPALA AND DISRUPTED RADIO AND TELEVISION SERVICES, UGANDAN RADIO SAID MONDAY.

THE UGANDA BROADCAST GAVE NO NAMES OR NUMBERS, BUT SAID THEY WERE SUBVERSIVE ELEMENTS SENT IN BY TANZANIAN PRESIDENT JULIUS NYERERE AND UGANDAN EXILES LIVING IN TANZANIA.

AN OFFICIAL AT THE SUDANESE EMBASSY, SPEAKING BY TELEPHONE FROM KAMPALA, SAID PEOPLE IN THE CITY WERE GOING ABOUT THEIR BUSINESS AS USUAL, BUT THAT PARTS OF THE CITY WERE WITHOUT POWER.

SELECTED SKETCHY SCRIPT $ARREST

```
 SCO
    BINDINGS        (&SIDE1 SV154 &PERSON SV155)
    REQUESTS        (R1 RQ79)
    INSTANCE        $ARREST
REQUESTS:
```

```
RQ79
 SATISFIED = T
    ((ACTOR &SIDE1 <=> (*ATRANS*) OBJECT (*CONT* TYPE
     (*SOCIAL*) PART &PERSON) TO &SIDE1 FROM &SIDE2))

SV154
      WORD#     (29)
      CERTAINTY          (10)
      TYPE      (*POLITY*)
      CONENT    (*UGANDA*)
SV155
      WORD#     (33)
      CERTAINTY          (10)
      TYPE      (*HUMAN*)
      CONENTCONEMOD    *UNSPEC*
      TYPE      (*SABOTEUR*)
      CONEHEAD *SABOTEUR*
      COMPOSITE          AMOUNT
      CONENT    (COMP0)

CPU TIME FOR UNDERSTANDING = 4634 MILLISECONDS

ENGLISH SUMMARY:
      UGANDA HAS ARRESTED SABOTEURS.
```

At this point, FRUMP did not have a sketchy script to process terrorist acts. It did, however, have a sketchy script to process arrests. Since there an arrest reported of the terrorists, FRUMP picked out that interpretation of the text.

The only facts that FRUMP could understand from this story are the identities of the authorities and the fact that they arrested saboteurs. FRUMP missed the nationality of the saboteurs. This is because the nationality is not explicitly stated. Rather it is an inference from the fact that the saboteurs were "SENT IN FROM TANZANIA." To understand that phrase, FRUMP must be able to predict that the event representing the saboteurs change in location is important in this situation. Of course, that event is not important in this situation. Rather it is important in the sketchy script to process terrorist acts. If FRUMP had such a script while reading the article, that script could have provided the information necessary for FRUMP to discover the saboteurs nationality.

UPI Story 3, March 15, 1979
INPUT:

JERUSALEM (UPI)-DEFENSE MINISTER EZER WEIZMAN FLEW TO THE UNITED STATES TODAY TO PIN DOWN AMERICAN PROMISES ACCOMPANYING THE EGYPT-ISRAEL PEACE TREATY, BUT ACKNOWLEDGED THE PACT IS BUT THE FIRST STEP TOWARD A REAL PEACE.

"THERE'S NO DOUBT THAT WE HAVE MANY, MANY PROBLEMS ON THE WAY TO A LIFE TOGETHER WITH OUR NEIGHBORS," WEIZMAN TOLD REPORTERS AT BEN GURION AIRPORT AS HE LEFT FOR WASHINGTON.

"BUT FIRST OF ALL, WE HAVE AN OPPORTUNITY THAT WE HAVE BEEN DREAMING OF FOR MANY YEARS-TO FIND A COMMON LANGUAGE WITH THE EGYPTIANS. AND THAT IN ITSELF IS NOT EASY," THE DEFENSE MINISTER SAID.

WEIZMAN WAS TO CONCLUDE AGREEMENTS WITH WASHINGTON ON THE MILITARY ANNEX TO THE PROPOSED TREATY AND FOR LONGTERM ARMS SUPPLIES TO ISRAEL AND FINANCIAL AID FOR PULLING ITS ARMY OUT OF THE SINAI AND REDEPLOYING IT IN THE NEGEV.

THE COST OF THE AID HAS BEEN ESTIMATED AT $4 BILLION TO $5 BILLION BY SEN. HOWARD BAKER, R-TENN., THOUGH SOME ISRAELI SOURCES HAVE PEGGED THE DIRECT AND INDIRECT IMPACT OF THE AID PACKAGE AT UP TO $10 BILLION.

THE ISRAELI DEFENSE MINISTER SAID HE WOULD ALSO BE MEETING WITH EGYPTIAN DEFENSE MINISTER KAMEL HASSAN ALI TO SETTLE OUTSTANDING PROBLEMS ON THE MILITARY ANNEX TO THE PEACE TREATY.

ASKED IF PROBLEMS REMAINED ON THE ANNEX, WEIZMAN SAID "THERE ARE NO PROBLEMS. THERE ARE TECHNICAL DETAILS, SOME SMALLER, SOME LARGER" INVOLVING THE ISRAELI MILITARY WITHDRAWAL FROM THE SINAI.

FOREIGN MINISTER MOSHE DAYAN, WHO WAS ORIGINALLY SCHEDULED TO ACCOMPANY WEIZMAN TO WASHINGTON, CANCELED OUT AT THE LAST MOMENT BUT MAY RUSH OVER IF HIS PRESENCE IS REQUIRED, OFFICIAL SOURCES SAID. WEIZMAN WAS EXPECTED TO WRAP UP HIS MISSION BEFORE SUNDAY, WHEN PRIME MINISTER MENACHEM BEGIN HAS SAID HIS CABINET WILL VOTE ON THE TREATY TEXT. THE TREATY WILL THEN GO TO THE PARLIAMENT, PROBABLY SUNDAY OR MONDAY.

UNOFFICIAL SURVEYS INDICATED WEDNESDAY THAT PARLIAMENT WILL ENDORSE THE TREATY BY A TWOTHIRDS MAJORITY, WITH OPPOSITION LIKELY ONLY FROM THE COMMUNISTS AND HAWKS WITHIN BEGIN'S OWN LIKUD BLOC AND ITS MAJOR COALIATION PARTNER, THE NATIONAL RELIGIOUS PARTY.

SELECTED SKETCHY SCRIPT $MEET

```
      BINDINGS
 (&HEAD1 G0468 &HEAD2 G0469 &TOPIC G0470 &DEST G0471
  &ORIG G0472 &TIME G0473 &HEAD3 G0474 &HEAD4 G0475
  &HEAD5 G0476)
      REQUESTS          (R1 G0465 R2 G0466 R3 G0467)
      INSTANCE          $MEET
REQUESTS:
 G0465
  SATISFIED = T
    ((<=> ($MEET MEETER &HEAD1 MEETEE &HEAD2 MTOPIC &TOPIC))
      LOC &DEST TIME &TIME)
 G0466
  SATISFIED = T
    ((ACTOR &HEAD3 <=> (*PTRANS*) OBJECT &HEAD3 TO &DEST
      FROM &ORIG) MODE (*POS*) TIME &TIME)

 G0468
      WORD#     (277)
      CERTAINTY         (10)
      TYPE      (*POLITY*)
      CONENT    (*ISRAEL*)
 G0469
      WORD#     (288)
      CERTAINTY         (10)
      TYPE      (*POLITY*)
      CONENT    (*EGYPT*)
 G0470
 G0471
      WORD#     (45)
      CERTAINTY         (10)
      TYPE      (*LOC-DEF*)
      CONENT    (*USA*)
 G0472
      WORD#     (0)
      CERTAINTY         (7)
      TYPE      (*LOC-DEF*)
      CONENT    (*ISRAEL*)
 G0473
 G0474
      WORD#     (42)
      CERTAINTY         (10)
      TYPE      (*PERSON-FRAME*)
      CONENT    *WEIZMAN*
      REPRESENT (G0468)

 G0475
 G0476

CPU TIME FOR UNDERSTANDING = 6173 MILLISECONDS
```

ENGLISH SUMMARY:
EZER WEIZMAN HAS GONE FROM ISRAEL TO THE UNITED STATES FOR A MEETING BETWEEN ISRAEL AND EGYPT.

FRUMP missed the emphasis of this story. Weizman went to the United States primarily to discuss the American role in Mid-East peace plans. However, as reported in paragraph six he also met with an Egyptian representative:

THE ISRAELI DEFENSE MINISTER SAID HE WOULD ALSO BE MEETING WITH EGYPTIAN DEFENSE MINISTER KAMEL HASSAN ALI TO SETTLE OUTSTANDING PROBLEMS ON THE MILITARY ANNEX TO THE PEACE TREATY.

This is the meeting that FRUMP picked up on. Since FRUMP missed the main point of the visit, it elevated the meeting with the Egyptian representative to the reason for the trip.

FRUMP did not understand "American promises" as the motivation behind the trip because FRUMP's current knowledge about peace treaties does not allow for third party mediators. The only way a country can relate to a peace treaty, as far as FRUMP is concerned, is by signing it or rejecting it.

It would be easy to add the fact that third party negotiators might exist in peace negotiations and that the principals might consult with them. It would, however, be very difficult for FRUMP to understand how the third party mediators interact and influence negotiations (i.e. what promises are likely to be made). This is due to limitations on the representational power of sketchy scripts. The behavior of mediators is not rigidly defined. They do what they perceive to be necessary to achieve peace. Their actions are highly dependent on each particular conflict situation. Since they are not stylized, these actions do not lend themselves well to script representation. Thus FRUMP might be modified to understand that Weizman came to discuss U.S. assurances. It would be difficult for FRUMP to understand exactly what assurances were discussed.

Notice that FRUMP inferred that Weizman came from Israel. This is not stated in the text. One of FRUMP's inference rules (in the CD Role Inferencer) states that the default place to start a trip from is ones home. In conceptual dependency terms this says that the FROM role of a PTRANS act may be filled with the home of the OBJECT filler if that information is available and the FROM role cannot be filled via a more certain rule.

```
UPI Story 4, December 5, 1978
INPUT:

     COLORADO (UPI)-RESCUERS ON SNOWCATS PUSHED THROUGH
FIVE-FOOT SNOWDRIFTS TO REACH THE WRECKAGE OF A TWINENGINE
ROCKY MOUNTAIN AIRWAYS PLANE THAT CRASHED IN COLORADO'S
RUGGED BUFFALO PASS.  AUTHORITIES REPORTED ONE PERSON HAD
BEEN KILLED BUT THAT 21 OTHERS ABOARD SURVIVED.  AMBULANCES
WERE ORDERED TO REMOVE THEM FROM THE WILDERNESS.

SELECTED SKETCHY SCRIPT $VEHICLE-ACCIDENT

       BUNDLE    (BN1)
       BINDINGS          (&OBJ1 SV164 &OBJ2 SV165 &LOC SV166)
       REQUESTS          (R0 RQ87 R1 RQ88)
       INSTANCE          $VEHICLE-ACCIDENT
REQUESTS:
 RQ88
  SATISFIED = T
     ((ACTOR &OBJ1 <=> (*PROPEL*) OBJECT &OBJ2) LOC &LOC
      MANNER (*VIOLENT*))

 SV164
       WORD#     (69)
       CERTAINTY         (10)
       TYPE      (*VEHICLE*)
       CONENT    (*PLANE*)
 SV165
       WORD#     (0)
       CERTAINTY         (6)
       TYPE      (*PHYSOBJ*)
       CONENT    (*GROUND*)
 SV166
       WORD#     (74)
       CERTAINTY         (10)
       TYPE      (*LOC-DEF*)
       CONENT    (*COLORADO*)

 ***BUNDLES:
 BN1
       BINDINGS
 (&DEADGRP SV167 &HURTGRP SV168 &MISSINGGRP SV169)
       REQUESTS          (R1 RQ89 R2 RQ90 R3 RQ91)
       INSTANCE          |CASUALTY
REQUESTS:
 RQ89
  SATISFIED = T
     ((ACTOR &DEADGRP IS (*HEALTH* VAL (-10))))

 SV167
       WORD#     (87)
       CERTAINTY         (10)
```

title officer does not conflict with this occupation, and


```
      TYPE      (*HUMAN*)
      AMOUNT    1
      CONENT    (COMPO)

SV168
SV169

CPU TIME FOR UNDERSTANDING = 2470 MILLISECONDS

ENGLISH SUMMARY:
      A VEHICLE ACCIDENT OCCURRED IN COLORADO.  A PLANE HIT
 THE GROUND.  1 PERSON DIED.
```

This story describes a plane crash in which one person was killed. The day this story was read FRUMP was not working with its standard English generator (the module that produces the English summaries from the conceptual representations). Instead it had a earlier simpler generator. While the summary lacks something in literary style, it does illustrate very well the underlying conceptual representation. In a vehicle accident, two of the important facts are the identities of the vehicle and physical object with which it collided. In this story, the vehicle is a plane and the physical object is the ground. FRUMP had to infer that the plane struck the ground. It was not stated in the text. The input only said that a plane crashed; it did not state into what. There is an inference rule in the CD Role Inferencer that states if an aircraft PROPELs something VIOLENTly, the object PROPELed is probably the ground. Notice that the script variable for the object PROPELed is bound to ground with only certainty 6.

This story also illustrates another point. It has probably occurred to the reader that if we were actually interested in producing summaries of stories for their own sake, rather than using summaries as a vehicle to study natural language, an approach might be simply to parrot back the first sentence of a news article. In this case we would get the bombastic summary:

RESCUERS ON SNOWCATS PUSHED THROUGH FIVE-FOOT SNOWDRIFTS TO REACH THE WRECKAGE OF A TWINENGINE ROCKY MOUNTAIN AIRWAYS PLANE THAT CRASHED IN COLORADO'S RUGGED BUFFALO PASS.

This "summary" leaves out the fact that anyone was killed and includes much irrelevant information such as the snow drifts were five-feet deep.

UPI Story 5, November 17, 1978
INPUT:

MOSCOW (UPI)-SOVIET PRESIDENT LEONID BREZHNEV TOLD A GROUP OF VISITING U. S. SENATORS FRIDAY THAT THE SOVIET UNION HAD ONCE "TESTED BUT NEVER STARTED PRODUCTION" OF A NEUTRON BOMB, BUT ONE SENATOR SAID HE DID NOT CONSIDER THE STATEMENT "A SERIOUS MATTER."

SEN. SAM NUNN, D-GA, WHO ATTENDED THE SESSION WITH BREZHNEV, COMMENTED ON THE NEUTRON BOMB STATEMENT TO REPORTERS AFTERWARD:

"THE WHOLE THING ABOUT WHETHER THEY TESTED ONE AND WHETHER THEY MIGHT HAVE THE CAPABILITY OF ONE IS NOT A VERY IMPORTANT FACTOR. IN SOME WAYS I WOULD FEEL MORE COMFORTABLE IF THE SOVIETS HAD NEUTRON WEAPONS RATHER THAN THE MONSTER THEY CALL TACTICAL NUCLEAR WEAPONS."

NEUTRON BOMBS WOULD BE USED LOCALLY FOR DEFENSE, PRIMARILY AGAINST A TANK ATTACK, AND ARE NOT CONSIDERED LONG-RANGE STRATEGIC WEAPONS.

SEN. ABRAHAM RIBICOFF, R-CONN, HEAD OF THE AMERICAN DELEGATION, SAID THE 65-MINUTE SESSION WITH BREZHNEV WAS A 90-DEGREE SHIFT FROM THE SHARP EXCHANGES BETWEEN THE SENATORS AND SOVIET PREMIER ALEXEI KOSYGIN ON THURSDAY.

SELECTED SKETCHY SCRIPT $FIGHTING

```
      BINDINGS
 (&SIDE0 SV163 &SIDE1 SV164 &SIDE2 SV165 &SIDE3 SV166
  &SIDE4 SV167 &SIDE5 SV168 &LSIDE SV169 &WSIDE SV170)
      REQUESTS          (R0 RQ84 R1 RQ85 R2 RQ86 R3 RQ87)
      INSTANCE          $FIGHTING
REQUESTS:
 RQ86
  SATISFIED = T
     ((ACTOR (*BOMBER* PART &SIDE4) <=> (*PTRANS*)
       OBJECT (*BOMB*)) LOC &SIDE5)

 SV163
 SV164
 SV165
 SV166
 SV167
      WORD#     (28)
      CERTAINTY          (10)
      TYPE      (*POLITY*)
      CONENT    (*USSR*)
 SV168
      WORD#     (7)
```

```
      CERTAINTY          (3)
      TYPE       (*POLITY*)
      CONENT     (*USSR*)
SV169
SV170
```

CPU TIME FOR UNDERSTANDING = 7887 MILLISECONDS

ENGLISH SUMMARY:
RUSSIAN BOMBERS HAVE ATTACKED RUSSIA.

There are a number of things wrong with FRUMP's understanding of this story. This is not a story about a war; it is about a bomb test. FRUMP decided there were bombers that dropped a bomb on Russia. It did this by interpreting the word "bomb" in the first sentence as a verb rather than a noun. Notice that while FRUMP is quite sure that Russia is the country that exploded the bomb, it is not so sure (certainty 3) that Russia is the target. That script variable was satisfied by parse rule 4. Recall that parse rule 4 does not require any syntactic confirmation for interpreting the text. It relies only on semantic properties of the words.

The reason FRUMP did not correctly understand this story is that it did not have the proper sketchy script. FRUMP built a conceptualization that was very close to the correct one. After all, a bomb was exploded (although it was more likely an underground explosion rather than dropped by bombers). When this story was processed, FRUMP had only one script that could account for bombings: the sketchy script to process war stories. FRUMP did not know that bombs could be exploded for other reasons (e.g. as a test of a weapon). This script was added along with the required vocabulary and FRUMP was asked to process the story again:

UPI Story 5, Version 2
INPUT:

MOSCOW (UPI)-SOVIET PRESIDENT LEONID BREZHNEV TOLD A GROUP OF VISITING U. S. SENATORS FRIDAY THAT THE SOVIET UNION HAD ONCE "TESTED BUT NEVER STARTED PRODUCTION" OF A NEUTRON BOMB, BUT ONE SENATOR SAID HE DID NOT CONSIDER THE STATEMENT "A SERIOUS MATTER."

SEN. SAM NUNN, D-GA, WHO ATTENDED THE SESSION WITH BREZHNEV, COMMENTED ON THE NEUTRON BOMB STATEMENT TO REPORTERS AFTERWARD:

"THE WHOLE THING ABOUT WHETHER THEY TESTED ONE AND WHETHER THEY MIGHT HAVE THE CAPABILITY OF ONE IS NOT A VERY IMPORTANT FACTOR. IN SOME WAYS I WOULD FEEL MORE COMFORTABLE IF THE SOVIETS HAD NEUTRON WEAPONS RATHER THAN THE MONSTER THEY CALL TACTICAL NUCLEAR WEAPONS."

NEUTRON BOMBS WOULD BE USED LOCALLY FOR DEFENSE, PRIMARILY AGAINST A TANK ATTACK, AND ARE NOT CONSIDERED LONG-RANGE STRATEGIC WEAPONS.

SEN. ABRAHAM RIBICOFF, R-CONN, HEAD OF THE AMERICAN DELEGATION, SAID THE 65-MINUTE SESSION WITH BREZHNEV WAS A 90-DEGREE SHIFT FROM THE SHARP EXCHANGES BETWEEN THE SENATORS AND SOVIET PREMIER ALEXEI KOSYGIN ON THURSDAY.

```
SELECTED SKETCHY SCRIPT $COMMENT

      BINDINGS
 (&SPEAKER SV170 &BRUNT SV171 &TOPIC SV172 &RECIP SV173)
      REQUESTS          (R1 RQ86)
      INSTANCE          $COMMENT
REQUESTS:
 RQ86
  SATISFIED = T
     ((ACTOR &SPEAKER <=> (*MTRANS*) MOBJECT (*CONCEPT*
       ACTION &TOPIC TYPE (*UNSPEC*) ABOUT &BRUNT)
       TO &RECIP) MANNER (*UNSPEC*))

 SV170
      WORD#     (13)
      CERTAINTY         (10)
      TYPE      (*POLITY*)
      CONENT    (*BREZHNEV*)
 SV171
 SV172
      TYPE      (*CD*)
      CONENT((<=>($TEST ACTOR (*USSR* CERTAINTY (10)
              WORD# (28)) OBJECT (*BOMB* CERTAINTY (10)
              WORD# (44))))

 SV173
      WORD#     (19)
      CERTAINTY         (10)
      TYPE      (*POLITY*)
      CONENT    (*USA*)


CPU TIME FOR UNDERSTANDING = 9531 MILLISECONDS

ENGLISH SUMMARY:
      LEONID BREZHNEV TOLD THE UNITED STATES THAT RUSSIA
 HAS TESTED A BOMB.
```

Here, FRUMP correctly understood the "bombing" as a weapons test. FRUMP now correctly identifies the verb of the phrase as "tested." "Bomb" is then interpreted as a noun rather than a verb.

UPI Story 6, March 15, 1979
INPUT:

ISRAELI-OCCUPIED WEST BANK (UPI)-ISRAELI SOLDIERS FIRED AT ROCK- THROWING PALESTINIANS PROTESTING THE EGYPTIAN-ISRAELI PEACE TREATY THURSDAY, KILLING A YOUNG MAN AND A TEEN-AGED GIRL AND WOUNDING A THIRD YOUTH.

IN THE MOST VIOLENT DAY OF PROTESTS YET IN THE OCCUPIED WEST BANK, SOLDIERS ALSO USED TEAR GAS AND FORCE TO DISPERSE DEMONSTRATORS IN EAST JERUSALEM, HEBRON, BETHLEHEM AND THE LARGELY CHRISTIAN SUBURB OF BEIT JALLAH, AS WELL AS JERICHO, RAMALLAH AND NEARBY REFUGEE CAMPS.

MILITARY SOURCES SAID THE SHOOTING AT HALHUL BEGAN WHEN A CROWD OF 600 HIGH SCHOOL STUDENTS AND ADULTS SURGED ONTO THE MAIN ROAD LINKING HEBRON AND JERUSALEM. THEY STONED A SQUAD OF EIGHT ISRAELI SOLDIERS AND A CAR CARRYING FOUR CIVILIANS.

AN OFFICIAL ARMY STATEMENT REPORTED THE SOLDIERS "WERE CAUGHT IN A VIOLENT DISTURBANCE." IT SAID BOTH CIVILIANS AND SOLDIERS FIRED WEAPONS, BUT DID NOT SAY WHO SHOT FIRST.

MAYOR MUHAMAD MILHAM SAID THE DEAD WERE NASRI HANANI, 21 A LABORER, AND RABA SHALALDE, 17 A HIGH SCHOOL STUDENT.

```
 SELECTED SKETCHY SCRIPT $DEMONSTRATION


      BINDINGS
 (&SIDE1 G0711 &SIDE2 G0712 &OBJ G0713 &PLCE1 G0714
 &PLCE2 G0715 &NAT1 G0716 &INVOLVE G0717 &EVENT G0718
 &COUNTRY G0719 &VICTIM G0720 &LOC G0721 &ACTION G0722)
      REQUESTS
 (R0 G0705 R1 G0706 R2 G0707 R3 G0708 R4 G0709 R6 G0710)
      INSTANCE          $DEMONSTRATION
REQUESTS:
 G0708
  SATISFIED = T
     ((ACTOR &SIDE1 NAT &NAT1 <=> (*MTRANS*) MOBJECT
      (*CONCEPT* ACTION &ACTION TYPE (*DISSENT*))))

 G0711
      WORD#     (47)
```

```
      CERTAINTY        (10)
      TYPE       (*HUMAN*)
      CONENT     (*PALESTINIAN*)
      AMOUNT     (G0699)

G0712
G0713
G0714
G0715
G0716
      WORD#      (0)
      CERTAINTY        (6)
      TYPE       (*POLITY*)
      CONENT     (*ISRAEL*)
G0717
G0718
G0719
G0720
G0721
G0722
```

CPU TIME FOR UNDERSTANDING = 7246 MILLISECONDS

ENGLISH SUMMARY:
ISRAELI PALESTINIANS HAVE VOICED DISSENT.

FRUMP correctly classified story 6 as a demonstration story. However, it missed the casualties. Notice that the nationality of the demonstrators is bound to Israel with only a certainty of 6. This is because the nationality had to be inferred from the location. FRUMP's CD Role Inferencer has a rule that if someone's nationality is needed and cannot be filled by a higer certainty process, his nationality is probably the same as the country of his location.

UPI Story 7, November 15, 1978
INPUT:

NEW YORK (UPI)-MARGARET MEAD, ONE OF THE NATION'S MOST FAMOUS ANTHROPOLOGISTS AND SOCIAL CRITICS, DIED WEDNESDAY OF CANCER AT THE AGE OF 76. DR. MEAD, WHO KNEW SHE WAS DYING, KEPT TO HER BUSY SCHEDULE OF RESEARCH AND WRITING UNTIL SHE WAS HOSPITALIZED SIX WEEKS AGO AT NEW YORK HOSPITAL.

A PRIVATE FUNERAL SERVICE AND BURIAL WAS BEING ARRANGED IN BUCKINGHAM, PA.

```
    THE 5-FOOT-2-INCH ANTHROPOLOGIST WAS A CLASSIC EXAMPLE
OF FEMALE LIBERATION YEARS BEFORE THE TERM BECAME POPULAR.
HER YEARS OF RESEARCH WITH PRIMITIVE AND SOPHISTICATED
CULTURES CONVINCED HER THAT MODERN SOCIETY MUST CURB ITS
AGGRESSIONS.

    IN AN INTERVIEW EARLIER THIS YEAR SHE SAID CHILDREN
WERE THE KEY TO THE UNITED STATES 'FUTURE- "WE HAVE TO LEARN
HOW TO LIVE WITHOUT VIOLENCE AND GREED."

SELECTED SKETCHY SCRIPT $DEATH


      BINDINGS          (&LOC SV157 &STIFF SV158)
      REQUESTS          (R1 RQ81)
      INSTANCE          $DEATH
REQUESTS:
 RQ81
  SATISFIED = T
     ((ACTOR &STIFF IS (*HEALTH* VAL (-10))) LOC &LOC)

 SV157
 SV158
      WORD#     (37)
      CERTAINTY         (10)
      TYPE      (*HUMAN-DEF*)
      CONENTSEX (FEMALE)
      NAME      (MEAD-0)
      FNAME     (MARGARET)
      LNAME     (MEAD)

CPU TIME FOR UNDERSTANDING = 5439 MILLISECONDS

ENGLISH SUMMARY:
      MARGARET MEAD DIED.
```

This story appeared on November 15, 1978. It illustrates FRUMP's ability to create a memory token for a person it does not know. Before FRUMP read this story it did not have "Margaret Mead" in its dictionary. It is impractical to attempt to anticipate every important person's name. Instead, we have given FRUMP heuristics on how to recognize people's names. When it has a top down prediction that an individual person (rather than a group) will be found at a specific location in the input, FRUMP will accept unknown text as a person's name if its heuristics are satisfied. Basically FRUMP's heuristics are that a name is a list of capitalized words perhaps with initials interspersed at the beginning and middle and optionally preceded by a title or occupation. FRUMP also has a list of typical first names in English. This list is

used to guess at the gender of the person. This is how FRUMP determined that Margaret Mead was a female.

When it recognizes a name, FRUMP creates a permanent memory token for the person where all the knowledge learned about that person is stored. In this example, the memory token created is MEAD-0. The information under MEAD-0 includes that it represents a definite person, the person is female with a first name of Margaret and a last name of Mead. If FRUMP had known the word "Anthropologist," it would have understood and stored away her occupation as well. FRUMP understood this story with $DEATH, the sketchy script to process obituaries. At the time this story was processed $DEATH did not contain the information that the person's age is important. Therefore, FRUMP could not add the information that she was 76.

tp 8 UPI Story 8, March 27, 1979
INPUT:

CLEVELAND (UPI)-EVER SINCE THE DAWN OF PANTYHOSE ON THE "UNMENTIONABLE" MARKET KILLED HER BUSINESS, GIRDLE DESIGNER PAULA BLATT HAS BEEN SCHEMING TO MAKE A COMEBACK. THE 78-YEAR -OLD WOMAN IS ATTEMPTING TO DESIGN A GIRDLE THAT WON'T SLIP AND SLIDE AND CAN BE WORN OVER PANTYHOSE. "SUCH A GIRDLE COULD REVITALIZE BUSINESSS THAT HAS BEEN LOST TO THE PANTYHOSE," SAID MRS. BLATT, HOLDER OF 52 PATENTS FOR HER CREATIONS-INCLUDING MATERNITY PANTIES, NO-CHAFE PANTIES AND GARTER BELTS.

AT ONE TIME, THE ACCLAIMED AUTHORITY ON MATERNITY LINGERIE AND COMFORT GARMENTS, SAID SHE WAS KNOWN AS "THE EXPECTANT MOTHER'S BEST FRIEND."

ALTHOUGH SHE NEVER HAD CHILDREN OF HER OWN, SHE STUDIED THE DISCOMFORTS OF CHILDBEARING AND, WITH THE HELP OF DOCTORS WHO APPROVED HER WORK, DESIGNED MATERNITY GARMENTS.

BUT HER BUSINESS, NU VOGUE CREATIONS, HAS NEVER HAD IT SO BAD.

IN 37 YEARS, SHE BUILT UP A CLIENTELE FROM HER DOWNTOWN SHOP WHICH ONCE INCLUDED SPECIALTY SHOPS ACROSS THE COUNTRY AND CUSTOMERS IN HONG KONG, EUROPE AND SOUTH AMERICA. BUT MACHINES IN HER SHOP WHICH ONCE HUMMED AS 15 WOMEN SEWED GIRDLES AND OTHER "UNMENTIONABLES" NOW ARE SILENT.

"MY SISTER LOANED ME A PORTABLE SEWING MACHINE AND I HAD TO SCRAPE UP $50 TO TO START A CHECKING ACCOUNT," SHE REMINISCED. "I HAD DIFFICULTY IN HEARING, I WAS TIMID AND I HATED SELLING, BUT I WAS FORCED TO MAKE A LIVING, SO I KEPT PLUGGING AWAY."

MRS. BLATT STILL COMES TO HER SHOP DAILY TO HELP HER TWO EMPLOYEES FILL ORDERS. SHE HOPES FOR THE DAY WHEN GIRDLES WILL COME BACK IN STYLE.

```
SELECTED SKETCHY SCRIPT $DEATH

 G0889
      BINDINGS          (&LOC G0891 &STIFF G0892)
      REQUESTS          (R1 G0890)
      INSTANCE          $DEATH
REQUESTS:
 G0890
  SATISFIED = T
     ((ACTOR &STIFF IS (*HEALTH* VAL (-10))) LOC &LOC)

 G0891
 G0892
      WORD#     (50)
      CERTAINTY         (10)
      TYPE      (*HUMAN-DEF*)
      CONENTSEX (FEMALE)
      NAME      (BLATT-0)
      FNAME     (PAULA)
      LNAME     (BLATT)

CPU TIME FOR UNDERSTANDING = 3287 MILLISECONDS

ENGLISH SUMMARY:
      PAULA BLATT HAS BEEN KILLED.
```

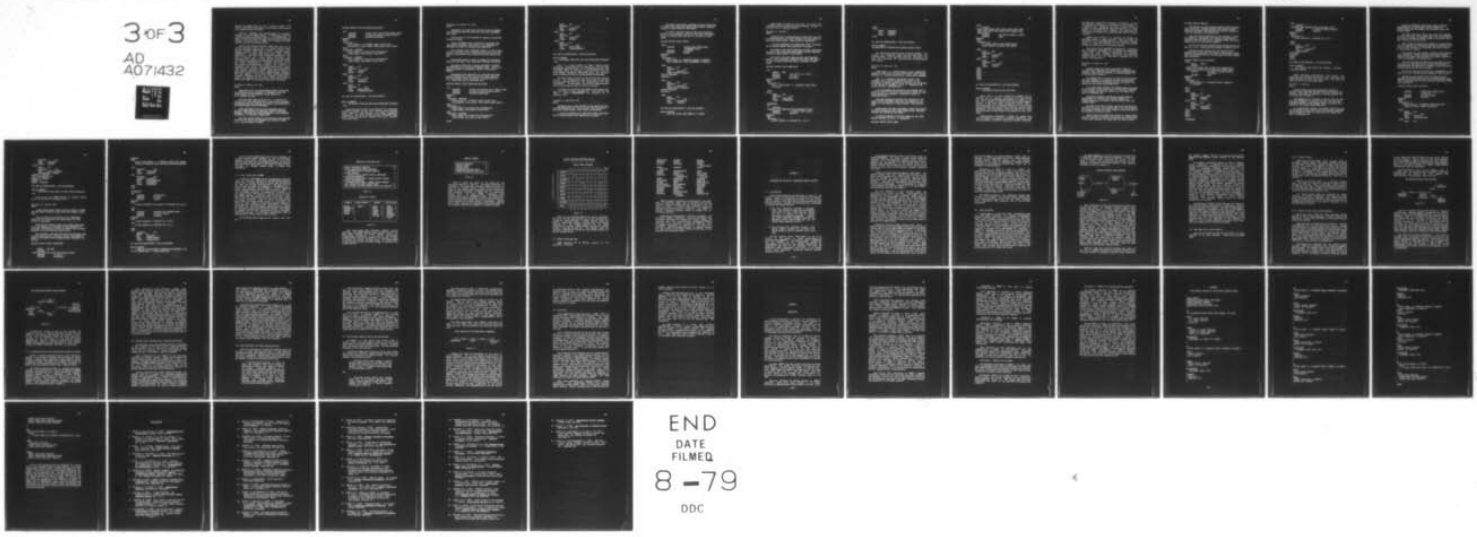
FRUMP completely misunderstood this story. It saw only the English input " XXXX KILLED XXXX PAULA BLATT." Since it could not interpret any of the intervening words, FRUMP thought it was about a woman dying.

This illustrates a very troublesome kind of error from which FRUMP suffers. To understand this story correctly FRUMP must be able to reject this interpretation. This interpretation can be eliminated by either of two methods: 1) give FRUMP the correct script so it will process the story correctly thus eliminating the false alarm 2) reject this interpretation on syntactic grounds - after all "PAULA BLATT" is not the object of the verb "KILLED," "BUSINESS" is.

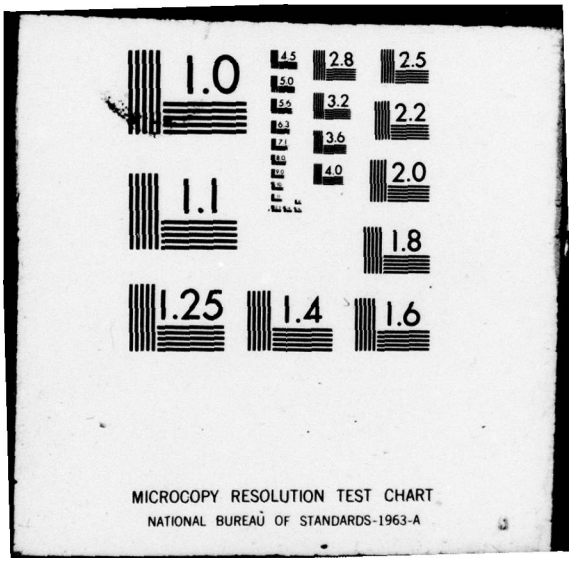
This story cannot be fixed by adding a sketchy script to FRUMP's repertoire. If the story is interesting, it is because it portrays a likable old woman in a rags-to-riches story. Perhaps it is also interesting because the idea of mentioning womens' "UNMENTIONABLES" seems humorous in our culture. At any rate the important points that ought to

AD-A071 432 YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE F/G 5/2
SKIMMING STORIES IN REAL TIME: AN EXPERIMENT IN INTEGRATED UNDE--ETC(U)
MAY 79 G F DEJONG N00014-75-C-1111
UNCLASSIFIED RR-158 NL

3 OF 3
AD
A071432



END
DATE
FILMED
8-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

show up in a summary are not of a scripty nature. The events described in this story are not easily anticipatable by a person writing the script.

However, rejecting the interpretation on syntactic grounds is not the answer either. If FRUMP were to perform a syntactic parse of the input prior to understanding, it would become very similar to many previous natural language systems. The whole point of FRUMP is to integrate the text analysis process with understanding.

What is the correct solution? FRUMP must not do an initial syntactic parse, and its script applier cannot provide the proper predictions to process the story. The correct solution would seem to be to augment FRUMP's PREDICTOR so it could make more than just scripty predictions. The flaw is not in FRUMP's organization. The flaw is relying on a script applier to reject this interpretation. Since a script applier cannot understand the correct interpretation of the article, it cannot reject the current interpretation. However, this is not to say that predictive understanders in general are incapable of processing this story; only script appliers are. A more flexible type of predictive understander, such as Wilensky's plan applier (Wilensky [1978]), might well be capable of making the required predictions to guide parsing. Currently the plan applier is not nearly robust enough for FRUMP's purposes. Presumably one day the problems prohibiting a robust plan applier will be solved. Until then we must be content for FRUMP to mis-process these kinds of stories.

UPI Story 9, February 16, 1979
INPUT:

WASHINGTON (UPI)-THE UNITED STATES FORMALLY RECOGNIZED THE AYATOLLAH RUHOLLAH KHOMEINI'S GOVERNMENT FRIDAY AND WARNED MOSCOW TO STOP SPREADING "FALSE ACCOUNTS OF U. S. ACTIONS" IN IRAN.

THE TWIN MOVES CAME ON THE EVE OF U. S. ATTEMPTS TO AIRLIFT UP TO 5,000 AMERICANS OUT OF IRAN WITHOUT BLOODSHED, AN OPERATION THAT WILL REQUIRE THE SUPPORT AND PROTECTION OF THE REVOLUTIONARY ISLAMIC GOVERNMENT.

STATE DEPARTMENT OFFICIALS SAID AMBASSADOR WILLIAM SULLIVAN, WHOSE EMBASSY WAS SACKED BY LEFTIST GUERRILLAS WEDNESDAY, DELIVERED THE TEHRAN GOVERNMENT A NOTE EXTENDING IT FORMAL U. S. DIPLOMATIC RECOGNITION.

THAT STEP SNAPPED AMERICA'S LAST OFFICIAL LINK WITH THE DYNASTY OF THE EXILED SHAH MOHAMMED REZA PAHLAVI-A PRIME U. S. ALLY FOR NEARLY 40 YEARS.

```
SELECTED SKETCHY SCRIPT $ESTABLISH-RELATIONS

 SC14
       BINDINGS          (&SIDE1 SV207 &SIDE2 SV208 &LEVEL SV209)
       REQUESTS          (R1 RQ104 R2 RQ105 R3 RQ106 R4 RQ107)
       INSTANCE          $ESTABLISH-RELATIONS
REQUESTS:
 RQ104
  SATISFIED = T
     ((ACTOR &SIDE1 <=> (*INVOKE*) MEANS (*LINK* TYPE
      (*DIPLOMATIC*) LEVEL &LEVEL) WITH &SIDE2) MODE (*POS*))
 RQ105
  SATISFIED = INFERRED
     ((ACTOR &SIDE1 IS (*LINK* TYPE (*DIPLOMATIC*)
       LEVEL &LEVEL) WITH &SIDE2) TIME (*TS*))
 RQ106
  SATISFIED = INFERRED
     ((ACTOR &SIDE2 IS (*LINK* TYPE (*DIPLOMATIC*)
       LEVEL &LEVEL) WITH &SIDE1) TIME (*TS*))

 SV207
       WORD#      (37)
       CERTAINTY          (10)
       TYPE       (*POLITY*)
       CONENT     (*USA*)
 SV208
       WORD#      (76)
       CERTAINTY          (10)
       TYPE       (*POLITY*)
       CONENT     (*KHOMENI*)
 SV209
       WORD#      (0)
       CERTAINTY          (4)
       TYPE       (*DPL-LEVEL*)
       CONENT     (*AMBASSADORIAL*)


CPU TIME FOR UNDERSTANDING = 5855 MILLISECONDS


ENGLISH SUMMARY:
       THE UNITED STATES AND IRAN HAVE ESTABLISHED DIPLOMATIC
 TIES.
```

This story and the next one illustrate the advantages activating sketchy scripts from concepts rather than key words. Both of the stories are about diplomatic recognition. This story contains the word "RECOGNIZE" which might be used as a key word. However, this is not possible in the next story.

UPI Story 10, February 16, 1979
INPUT:

WASHINGTON (UPI)-THE UNITED STATES IS READY TO RESUME FULL DIPLOMATIC RELATIONS WITH IRAQ, THE STATE DEPARTMENT SAID TODAY.

IRAQ HAS BEEN IN THE FOREFRONT OF RADICAL ARAB NATIONS OPPOSED TO ISRAEL.

TODAY'S STATEMENT CAME IN RESPONSE TO OVERTURES FROM THE BAGHDAD GOVERNMENT, WHICH SAID IT WAS INTERESTED IN IMPROVING THE LEVEL OF RELATIONS WITH WASHINGTON.

BOTH COUNTRIES HAVE "INTERESTS SECTIONS" IN THE OTHER'S CAPITAL, BUT THEY HAVE NOT EXCHANGED AMBASSADORS SINCE IRAQ BROKE OFF RELATIONS IN 1967 DURING THE SIX-DAY MIDDLE EAST WAR.

"THE UNITED STATES IS READY TO RESUME FULL DIPLOMATIC RELATIONS WITH IRAQ," SAID DEPARTMENT SPOKESMAN TOM RESTON.

THE UNITED STATES HAS 15 PEOPLE STATIONED IN BAGHDAD AND OPERATING OUT OF THE BELGIAN EMBASSY THERE. IRAQ HAS 10 PEOPLE STATIONED IN WASHINGTON, WORKING IN THE INDIAN EMBASSY.

DISCUSSIONS ABOUT RESTORING FULL RELATIONS HAVE BEEN GOING ON SPORADICALLY SINCE 1975 BUT THE IRAQIS BROKE OFF THE TALKS SEVERAL TIMES, APPARENTLY IN PROTEST OVER THE U. S. MEDIATION ROLE IN THE MIDDLE EAST.

SELECTED SKETCHY SCRIPT $ESTABLISH-RELATIONS

```
 SC11
      BINDINGS          (&SIDE1 SV186 &SIDE2 SV187 &LEVEL SV188)
      REQUESTS          (R1 RQ94 R2 RQ95 R3 RQ96 R4 RQ97)
      INSTANCE          $ESTABLISH-RELATIONS
REQUESTS:
 RQ94
  SATISFIED = INFERRED
     ((ACTOR &SIDE1 <=> ('INVOKE') MEANS ('LINK' TYPE
      ('DIPLOMATIC') LEVEL &LEVEL) WITH &SIDE2) MODE ('POS'))
 RQ95
  SATISFIED = T
     ((ACTOR &SIDE1 IS ('LINK' TYPE ('DIPLOMATIC')
       LEVEL &LEVEL) WITH &SIDE2) TIME ('TS'))
 RQ96
  SATISFIED = INFERRED
     ((ACTOR &SIDE2 IS ('LINK' TYPE ('DIPLOMATIC')
       LEVEL &LEVEL) WITH &SIDE1) TIME ('TS'))

 SV186
```

```
      WORD#      (27)
      CERTAINTY          (10)
      TYPE       (*POLITY*)
      CONENT     (*USA*)
SV187
      WORD#      (41)
      CERTAINTY          (10)
      TYPE       (*POLITY*)
      CONENT     (*IRAQ*)
SV188
      WORD#      (0)
      CERTAINTY          (4)
      TYPE       (*DPL-LEVEL*)
      CONENT     (*AMBASSADORIAL*)

CPU TIME FOR UNDERSTANDING = 6639 MILLISECONDS

ENGLISH SUMMARY:
      THE UNITED STATES AND IRAQ HAVE ESTABLISHED DIPLOMATIC
 TIES.
```

This is a prime example of event induced script activation. In the absence of a semantically rich word like "RECOGNIZE" FRUMP must build up a conceptualization from individual words to activate a sketchy script. Here the words "RESUME DIPLOMATIC RELATIONS" build the structure which activates the sketchy script $ESTABLISH-RELATIONS.

The same concept is built for each of the two previous stories. The difference is that in the first one the bulk of the concept was built from the semantically rich word "RECOGNIZE." In the second story an entire English phrase had to be analyzed in order to build the concept.

It should be pointed out that FRUMP did not decide that the ties were on the ambassadorial level from the word "FULL." Rather "AMBASSADORIAL" was supplied by an inference rule.

UPI Story 11, March 26, 1979
INPUT:

JERUSALEM (UPI)-A BOMB EXPLODED IN THE WALLED OLD CITY OF JERUSALEM MONDAY JUST ABOUT THE SAME TIME THE LEADERS OF EGYPT AND ISRAEL SIGNED A PEACE TREATY IN WASHINGTON.

A POLICE SPOKESWOMAN SAID THERE WERE "SOME" CASUALTIES BUT NO FURTHER DETAILS WERE IMMEDIATELY AVAILABLE.

```
    THE BOMBING CAME DESPITE INCREASED SECURITY PRECAUTIONS
BY ISRAELI ARMY TROOPS AND BORDER GUARDS AGAINST POSSSIBLE
ATTACKS BY ARABS OPPOSED TO THE TREATY.

    BUT ARABS IN THE WEST BANK AND GAZA STRIP DECLARED
MONDAY A DAY OF MOURNING, SHUTTING THEIR SHOPS FROM NABLUS
TO GAZA TO PROTEST WHAT THEY SEE AS A BETRAYAL OF THEIR
CAUSE.

SELECTED SKETCHY SCRIPT $AGREE

 G0543
       BINDINGS            (&SIDE1 G0545 &SIDE2 G0546
                            &AGREEMENT G0547)
       REQUESTS            (R1 G0544)
       INSTANCE            $AGREE
REQUESTS:
 G0544
  SATISFIED = T
     ((ACTOR &SIDE1 <=> (*MTRANS*) MOBJECT (*CONCEPT*
       TYPE (*AGREEING*) ABOUT &AGREEMENT) TO &SIDE2))

 G0545
       WORD#      (75)
       CERTAINTY          (10)
       TYPE       (*POLITY*)
       CONENT(*EGYPT* *ISRAEL*)
(*EGYPT* *ISRAEL*)
(*ANIMATE*)

 G0546
       WORD#      (75)
       CERTAINTY          (10)
       TYPE       (*POLITY*)
       CONENT(*EGYPT* *ISRAEL*)
(*EGYPT* *ISRAEL*)
(*ANIMATE*)

 G0547
       WORD#      (79)
       CERTAINTY          (10)
       TYPE       (*MOBJECT*)
       CONENT     (*TREATY*)


CPU TIME FOR UNDERSTANDING = 2763 MILLISECONDS

ENGLISH SUMMARY:
       EGYPT AND ISRAEL HAVE AGREED TO A TREATY.
```

FRUMP missed the point of this story. It did not have a sketchy script to process terrorist bombings. However, it does have one for signing treaties.

```
UPI Story 11, Version 2
INPUT:

     JERUSALEM (UPI)-A BOMB EXPLODED 1N THE WALLED OLD CITY
OF JERUSALEM MONDAY JUST ABOUT THE SAME TIME THE LEADERS OF
EGYPT AND ISRAEL SIGNED A PEACE TREATY IN WASHINGTON.

     A POLICE SPOKESWOMAN SAID THERE WERE "SOME" CASUALTIES
BUT NO FURTHER DETAILS WERE IMMEDIATELY AVAILABLE.

     THE BOMBING CAME DESPITE INCREASED SECURITY PRECAUTIONS
BY ISRAELI ARMY TROOPS AND BORDER GUARDS AGAINST POSSSIBLE
ATTACKS BY ARABS OPPOSED TO THE TREATY.

     BUT ARABS IN THE WEST BANK AND GAZA STRIP DECLARED
MONDAY A DAY OF MOURNING, SHUTTING THEIR SHOPS FROM NABLUS
TO GAZA TO PROTEST WHAT THEY SEE AS A BETRAYAL OF THEIR
CAUSE.

SELECTED SKETCHY SCRIPT $EXPLOSION

 SC0
       BUNDLE     (BN0)
       BINDINGS            (&OBJ SV163 &LOC SV164)
       REQUESTS            (R1 RQ84)
       INSTANCE            $EXPLOSION
REQUESTS:
 RQ84
  SATISFIED = T
     ((ACTOR (*EXPLOSION*) <=> (*PROPEL*) OBJECT &OBJ)
       LOC &LOC)

 SV163
 SV164
       WORD#      (60)
       CERTAINTY           (10)
       TYPE       (*LOC-DEF*)
       CONENT     (*JERUSALEM*)

 ***BUNDLES:
 BN0
       BINDINGS
 (&DEADGRP SV165 &HURTGRP SV166 &MISSINGGRP SV167)
       REQUESTS            (R1 RQ85 R2 RQ86 R3 RQ87)
       INSTANCE            |CASUALTY
REQUESTS:
 RQ86
  SATISFIED = T
     ((ACTOR &HURTGRP IS (*HEALTH* VAL (-3))))
```

```
SV165
SV166
      AMOUNT    (*UNSPEC*)
      TYPE      (*HUMAN*)
      CONENT    (*HUMAN*)
SV167
```

CPU TIME FOR UNDERSTANDING = 5250 MILLISECONDS

ENGLISH SUMMARY:
AN EXPLOSION IN JERUSALEM HAS INJURED SEVERAL PEOPLE.

The appropriate sketchy script was added together with several new vocabulary words and the same story was processed again. This time FRUMP correctly classified the story and understood the important fact that some people were injured.

UPI Story 12, March 26, 1979
INPUT:

MIAMI (UPI)-U. S. SURGEON GENERAL JULIUS RICHMOND MET MONDAY FOR THREE HOURS IN HAVANA WITH CUBA'S PUBLIC HEALTH MINISTER, DR. JOSE GUITIERREZ MUNIZ, HAVANA RADIO ANNOUNCED.

THE 7 P. M. BROADCAST, MONITORED IN MIAMI, SAID RICHMOND, ALSO ASSISTANT SECRETARY FOR HEALTH IN THE DEPARTMENT OF HEALTH, EDUCATION AND WELFARE, WAS ACCOMPANIED TO THE MEETING BY "MEMBERS OF THE U. S. DELEGATION WHO ARRIVED IN CUBA A FEW HOURS AGO."

(AN HEW SPOKESMAN IN WASHINGTON SAID RICHMOND WENT TO HAVANA TO "LOOK AT THE HEALTH CARE DELIVERY SYSTEM IN CUBA." THE SPOKESMAN SAID RICHMOND IS EXPECTED TO RETURN TO WASHINGTON FRIDAY.)

THE CUBAN DELEGATION INCLUDED VICE MINISTERS OF THE PUBLIC HEALTH DEPARTMENT ALONG WITH "ALL MEMBERS OF THE COUNCIL OF THE MINISTRY OF PUBLIC HEALTH," THE BROADCAST SAID.

HAVANA RADIO SAID MUNIZ DELIVERED A "BROAD EXPLANATION OF HEALTH SERVICES IN CUBA, THEIR ORGANIZATION, PRINCIPAL PROBLEMS AND PROSPECTIVE SOLUTIONS."

IT QUOTED RICHMOND AS PRAISING CUBANS FOR THE "WARM WELCOME RECEIVED" BY THE U. S. DELEGATION.

SELECTED SKETCHY SCRIPT $MEET

```
G0834
      BINDINGS
(&HEAD1 G0838 &HEAD2 G0839 &TOPIC G0840 &DEST G0841
 &ORIG G0842 &TIME G0843 &HEAD3 G0844 &HEAD4 G0845
 &HEAD5 G0846)
      REQUESTS          (R1 G0835 R2 G0836 R3 G0837)
      INSTANCE          $MEET
REQUESTS:
 G0835
  SATISFIED = T
     ((<=> ($MEET MEETER &HEAD1 MEETEE &HEAD2
       MTOPIC &TOPIC)) LOC &DEST TIME &TIME)

 G0838
      WORD#     (28)
      CERTAINTY          (10)
      TYPE      (*POLITY*)
      CONENT    (*USA*)
 G0839
      WORD#     (50)
      CERTAINTY          (10)
      TYPE      (*POLITY*)
      CONENT    (*CUBA*)
 G0840
 G0841
 G0842
 G0843
 G0844
 G0845
 G0846

CPU TIME FOR UNDERSTANDING = 5403 MILLISECONDS

ENGLISH SUMMARY:
      THE UNITED STATES HAS MET WITH CUBA.
```

While this is a very superficial summary, it is adequate for the purpose the meeting script was designed for. One of the anticipated applications of FRUMP is as part of a system to predict world crises from newspaper stories. It has been shown (McClelland [1969]) that a statistical index can be computed between pairs of countries which correlates well with crisis situations between those countries. The statistical index is a function of recent actions between the two countries as reported in the world press.

McClelland has developed a scheme for coding news articles designed to capture the relevant information. Each code represents a particular interaction between countries.

The codes are classified as friendly or unfriendly. To take two extremes, giving aid is a friendly act while bombing is a hostile act. Basically the idea is that a crisis is unlikely to occur between two countries soon after a large number of friendly acts. However, the occurrence of unfriendly acts between two countries might lead to a crisis situation.

FRUMP's role in this is to automate the process of assigning codes to news articles. This has been the motivation for giving FRUMP most of the particular sketchy scripts it has and accounts for the heavy bias toward scripts for dealing with international events.

The sketchy script $MEETING was designed to understand the information necessary for producing the appropriate crisis code. This includes identifying the countries involved and that a meeting took place. Thus the above summary captures all the information necessary for the correct code generation.

UPI Story 13, March 15, 1979
INPUT:

NAIROBI, KENYA (UPI)-ARAB STATES HAVE PLEDGED $4 MILLION TO HELP PRESIDENT IDI AMIN BATTLE AN INVASION FROM TANZANIA, WHICH AMIN IS CONFIDENT OF CRUSHING BECAUSE "GOD IS ON THE SIDE OF UGANDA."

THE UGANDAN ARMY IS "WILLING TO FIGHT TO THE LAST MAN," AMIN SAID WEDNESDAY IN APPEALING FOR HELP FROM FOUR ARAB MINISTERS GATHERED IN THE UGANDAN CAPITAL OF KAMPALA FOR A MEETING OF THE ISLAMIC DEVELOPMENT BANK.

IN A CONCILIATORY GESTURE, THE UGANDAN LEADER ALSO SAID HE WAS WILLING TO HOLD PEACE TALKS WITH TANZANIAN PRESIDENT JULIUS NYERERE AND VOWED HIS TROOPS WILL NOT INVADE "ONE INCH" OF TANZANIAN SOIL.

TANZANIAN GOVERNMENT SOURCES HAVE ALREADY REJECTED SUGGESTIONS FOR A FACE-TO-FACE MEETING BETWEEN THE UNPREDICTABLE AMIN AND NYERERE, WHO HAS VOWED TO OUST THE UGANDAN LEADER.

THE ENVOYS FROM SAUDI ARABIA, THE UNITED ARAB EMIRATES, ABU DHABI AND LIBYA, WHICH ALREADY SENT 1,000 AND PLANELOADS OF MILITARY EQUIPMENT TO UGANDA, AGREED TO GIVE AMIN, WHO IS NOW A MOSLEM, $4 MILLION IN AID.

AMIN'S APPEALS COINCIDED WITH REPORTS BY UGANDAN EXILES THAT THE INVADERS HAD DEFEATED PALESTINIAN-LED UGANDAN UNITS IN THE LATEST FIGHTING IN CLASHES NEAR THE TOWN OF LUKAYA,

```
65 MILES SOUTH OF KAMPALA.

     THE SOURCES SAID THE UGANDANS RETREATED AFTER SUFFERING
HEAVY CASUALTIES.  WOUNDED SOLDIERS WERE JAMMING KAMPALA
HOSPITALS AND OTHERS WERE EVACUATED BY MILITARY AIRCRAFT TO
LIBYA FOR TREATMENT, THE SOURCES SAID.

     AMIN TOLD THE ARAB MINISTERS HE WAS CONFIDENT HIS ARMY
WAS CAPABLE OF "CONTAINING" THE INVASION AND OUSTING THE
ATTACKERS FROM THE LARGE AREA OF SOUTHERN UGANDA THEY OCCUPY
BECAUSE "GOD IS ON THE SIDE OF UGANDA."

     BUT HE ALSO SAID THE INVADING FORCE OUTNUMBERED HIS OWN
ARMY BY A 3-TO -1 MARGIN AND WAS COMPOSED OF SOME "50,000."

     WESTERN MILITARY SOURCES SAID THE INVASION FORCE
NUMBERED LITTLE MORE THAN 4,000 MOST OF THEM REGULAR TROOPS
FROM THE TANZANIAN ARMY BUT ALSO INCLUDING ANTI-AMIN UGANDAN
EXILE GUERRILLAS.

SELECTED SKETCHY SCRIPT $FIGHTING
 G0506
       BUNDLE     (G0521)
       BINDINGS
 (&SIDE0 G0511 &SIDE1 G0512 &SIDE2 G0513 &SIDE3 G0514
  &SIDE4 G0515 &SIDE5 G0516 &LSIDE G0517 &WSIDE G0518)
       REQUESTS          (R0 G0507 R1 G0508 R2 G0509
                          R3 G0510)
       INSTANCE          $WAR
REQUESTS:
 G0508
  SATISFIED = T
     ((ACTOR &SIDE2 <=> (*PTRANS*) OBJECT (*TROOPS*)
       TO &SIDE3))

 G0511
 G0512
 G0513
       WORD#      (51)
       CERTAINTY          (10)
       TYPE       (*POLITY*)
       CONENT     (*AMIN*)
 G0514
       WORD#      (57)
       CERTAINTY          (10)
       TYPE       (*POLITY*)
       CONENT     (*TANZANIA*)
 G0515
 G0516
 G0517
 G0518

***BUNDLES:
```

```
 G0521
       BINDINGS
 (&DEADGRP G0525 &HURTGRP G0526 &MISSINGGRP G0527)
       REQUESTS          (R1 G0522 R2 G0523 R3 G0524)
       INSTANCE          ¦CASUALTY
REQUESTS:
 G0523
  SATISFIED = T
     ((ACTOR &HURTGRP IS (*HEALTH* VAL (-3))))

 G0525
 G0526
       WORD#     (334)
       CERTAINTY         (10)
       TYPE      (*HUMAN*)
       CONENT*UNSPEC*
(*SOLDIER*)
*SOLDIER*
AMOUNT
(G0520)

CPU TIME FOR UNDERSTANDING = 7575 MILLISECONDS

ENGLISH SUMMARY:
       IDI AMIN HAS SENT TROOPS INTO TANZANIA.  SOLDIERS
 HAVE BEEN INJURED.
```

FRUMP completely misunderstood this story. The vocabulary word "PLEDGED" was not yet in the dictionary when FRUMP processed this story the first time. The word was added and the story was skimmed again.

UPI Story 13, Version 2
INPUT:

NAIROBI, KENYA (UPI)-ARAB STATES HAVE PLEDGED $4 MILLION TO HELP PRESIDENT IDI AMIN BATTLE AN INVASION FROM TANZANIA, WHICH AMIN IS CONFIDENT OF CRUSHING BECAUSE "GOD IS ON THE SIDE OF UGANDA."

THE UGANDAN ARMY IS "WILLING TO FIGHT TO THE LAST MAN," AMIN SAID WEDNESDAY IN APPEALING FOR HELP FROM FOUR ARAB MINISTERS GATHERED IN THE UGANDAN CAPITAL OF KAMPALA FOR A MEETING OF THE ISLAMIC DEVELOPMENT BANK.

IN A CONCILIATORY GESTURE, THE UGANDAN LEADER ALSO SAID HE WAS WILLING TO HOLD PEACE TALKS WITH TANZANIAN PRESIDENT JULIUS NYERERE AND VOWED HIS TROOPS WILL NOT INVADE "ONE INCH" OF TANZANIAN SOIL.

TANZANIAN GOVERNMENT SOURCES HAVE ALREADY REJECTED SUGGESTIONS FOR A FACE-TO-FACE MEETING BETWEEN THE UNPREDICTABLE AMIN AND NYERERE, WHO HAS VOWED TO OUST THE UGANDAN LEADER.

THE ENVOYS FROM SAUDI ARABIA, THE UNITED ARAB EMIRATES, ABU DHABI AND LIBYA, WHICH ALREADY SENT 1,000 AND PLANELOADS OF MILITARY EQUIPMENT TO UGANDA, AGREED TO GIVE AMIN, WHO IS NOW A MOSLEM, $4 MILLION IN AID.

AMIN'S APPEALS COINCIDED WITH REPORTS BY UGANDAN EXILES THAT THE INVADERS HAD DEFEATED PALESTINIAN-LED UGANDAN UNITS IN THE LATEST FIGHTING IN CLASHES NEAR THE TOWN OF LUKAYA, 65 MILES SOUTH OF KAMPALA.

THE SOURCES SAID THE UGANDANS RETREATED AFTER SUFFERING HEAVY CASUALTIES. WOUNDED SOLDIERS WERE JAMMING KAMPALA HOSPITALS AND OTHERS WERE EVACUATED BY MILITARY AIRCRAFT TO LIBYA FOR TREATMENT, THE SOURCES SAID.

AMIN TOLD THE ARAB MINISTERS HE WAS CONFIDENT HIS ARMY WAS CAPABLE OF "CONTAINING" THE INVASION AND OUSTING THE ATTACKERS FROM THE LARGE AREA OF SOUTHERN UGANDA THEY OCCUPY BECAUSE "GOD IS ON THE SIDE OF UGANDA."

BUT HE ALSO SAID THE INVADING FORCE OUTNUMBERED HIS OWN ARMY BY A 3-TO -1 MARGIN AND WAS COMPOSED OF SOME "50,000."

WESTERN MILITARY SOURCES SAID THE INVASION FORCE NUMBERED LITTLE MORE THAN 4,000 MOST OF THEM REGULAR TROOPS FROM THE TANZANIAN ARMY BUT ALSO INCLUDING ANTI-AMIN UGANDAN EXILE GUERRILLAS.

SELECTED SKETCHY SCRIPT $GIVE-AID

```
      BINDINGS          (&DONOR SV161 &RECIP SV162
                         &AID SV163)
      REQUESTS          (R1 RQ83 R2 RQ84)
      INSTANCE          $GIVE-AID
REQUESTS:
 RQ83
  SATISFIED = T
     ((ACTOR &DONOR <=> (*ATRANS*) OBJECT &AID FROM
       &DONOR TO &RECIP) MANNER (*WILLING*))

 SV161
      WORD#     (37)
      CERTAINTY         (10)
      TYPE      (*POLITY*)
      CONENT    (*ARAB-NATIONS*)
 SV162
      WORD#     (51)
```

```
       CERTAINTY          (10)
       TYPE       (*POLITY*)
       CONENT     (*AMIN*)
 SV163
       WORD#      (41)
       CERTAINTY          (10)
       TYPE       (*DOLLAR-VALUE*)
       CONENTCONEMOD = 4000000
CONEHEAD = *DOLLAR*
TYPE = (*DOLLAR-VALUE*)
COMPOSITE = AMOUNT
REMPROP = T
CONENT = (COMPOSO)

CPU TIME FOR UNDERSTANDING = 6273 MILLISECONDS

ENGLISH SUMMARY:
       ARAB NATIONS HAVE GIVEN IDI AMIN 4000000 DOLLARS IN
 AID.
```

With the new word, FRUMP selected the correct sketchy script and picked up the amount of aid.

```
UPI Story 14, June 20, 1978
INPUT:

     A SHARP EARTH TREMOR TUESDAY SHOOK BUILDINGS IN GREECE
AND SENT PEOPLE FLEEING IN PANIC FROM THEIR HOUSES, POLICE
SAID.

     THE FULL EXTENT OF THE DAMAGE WAS NOT IMMEDIATELY
KNOWN, BUT POLICE REPORTED THAT THREE PEOPLE HAD BEEN
INJURED IN THE QUAKE.

     THERE WAS NO IMMEDIATE REPORT FROM ATHENS OBSERVATORY
AS TO THE INTENSITY OF THE TREMOR BUT THE UNIVERSITY OF
CALIFORNIA SEISMOGRAPHIC STATION IN BERKELEY SAID THE QUAKE
MEASURED 6.3 ON THE OPEN-END RICHTER SCALE.

     THE CALIFORNIA LABORATORY PINPOINTED THE EPICENTER SOME
60 MILES SOUTH OF SOFIA, BULGARIA, AND 200 MILES NORTH OF
ATHENS IN THE BORDER REGION BETWEEN GREECE, YUGOSLAVIA AND
BULGARIA.

SELECTED SKETCHY SCRIPT $EARTHQUAKE


       BUNDLE     (BN1 BN2)
       BINDINGS
 (&LOC SV30 &MAG SV31 &TIME SV32 &DURATION SV33)
       REQUESTS           (R1 RQ14)
       INSTANCE           $EARTHQUAKE
```

```
REQUESTS:
 RQ14
    ((ACTOR (*GEO-FORCE*) <=> (*PTRANS*) OBJECT &LOC) MANNER
     (*CYCLIC*) MAGNITUDE &MAG TIME &TIME DURATION &DURATION)

 SV30
       CERTAINTY          (10)
       TYPE       (*LOCATION*)
       CONENT     (*GREECE*)
 SV31
       CERTAINTY          (10)
       READING    (6.3000000)
       SCALE      (*RICHTERSCALE*)
       TYPE       (*QUAKEMAG*)
       CONENT     (*QUAKEMAG*)
 SV32
 SV33

 ***BUNDLES:
 BN1
       BINDINGS           (&VALUE SV34)
       REQUESTS           (R1 RQ15)
       INSTANCE           |DAMAGE
REQUESTS:
 RQ15
     ((ACTOR (*PROPERTY* VAL &VALUE) IS (*PSTATE* VAL (-3))))

 SV34
 BN2
       BINDINGS           (&DEADGRP SV35 &HURTGRP SV36)
       REQUESTS           (R1 RQ16 R2 RQ17)
       INSTANCE           |CASUALTY
REQUESTS:
 RQ16
     ((ACTOR &DEADGRP IS (*HEALTH* VAL (-10))))
 RQ17
     ((ACTOR &HURTGRP IS (*HEALTH* VAL (-3))))

 SV35
 SV36
       CERTAINTY          (10)
       AMOUNT     (3)
       UNIT       (*PERSON*)
       TYPE       (*HUMAN-GROUP*)
       CONENT     (*HUMAN-GROUP*)

CPU TIME FOR UNDERSTANDING = 5904 MILLISECONDS

ENGLISH SUMMARY:
       THERE WAS AN EARTHQUAKE IN GREECE WHICH MEASURED 6.300
 ON THE RICHTER SCALE.  3 PEOPLE WERE HURT.
```

Story 14 again demonstrates that the first sentence of a news article cannot reliably be used as an adequate summary of the article. The first sentence of this story mentions that there was an earthquake in Greece but does not give the Richter scale level or the number of people killed. Obviously an adequate summary ought to contain that information.

7.3 A Day in the Life of FRUMP

On April 5, 1979 FRUMP was run continuously on UPI wire input for 24 hours. During this time 368 stories appeared on the wire. Of these 100 were not actual news articles and 147 were not scripty news articles. Thus there were 121 news stories that could in principle be understood by a script type approach such as FRUMP's. In fact, FRUMP had correct sketchy scripts for 29 of the articles of which 11 were processed correctly. As pointed out in chapter 1, on the average, 50% of the stories on the UPI wire are scripty, and FRUMP generally processes about 10% correctly. Here FRUMP correctly processed only about 3% of the total number of stories correctly. This is due to the comparatively small number of scripty stories on April 5. Only about one third of total number of stories were scripty compared to the normal 50%. Furthermore, the distribution of scripty situations was such that FRUMP's sketchy scripts accounted for less than a quarter of the scripty stories. This figure is usually close to 30%. Taking these facts into account, FRUMP did quite well. It understood approximately 38% of the stories for which it had a sketchy script.

The following tables summarize the results from that day.

ANALYSIS OF THE FRUMP RUN

STORIES UNDERSTOOD CORRECTLY	11
STORIES UNDERSTOOD ALMOST CORRECTLY (INCORRECT SCRIPT VARIABLE BINDING)	2
STORIES IGNORED (CORRECT SCRIPT PRESENT BUT NOT IDENTIFIED)	13
STORIES MISUNDERSTOOD (INCORRECT SCRIPT USED - CORRECT SCRIPT MISSING)	8
STORIES MISUNDERSTOOD (INCORRECT SCRIPT USED - CORRECT SCRIPT PRESENT)	1
TOTAL STORIES FOR WHICH SCRIPTS EXIST	29
TOTAL STORIES FOR WHICH CORRECT SCRIPT WAS SELECTED	13

Table 7.1

ANALYSIS BY SCRIPT

CORRECT		WRONG BINDING		IGNORED		FALSE ALARM	
$FIGHT	3	$MEET	2	$FIRE	4	$MEET	2
$COMMENT	3			$MEET	2	$REJECT	2
$THREAT	2			$FIGHT	2	$THREAT	1
$DEATH	2			$DEATH	2	$DEATH	1
$VEHAC	1			$VEHAC	2	$PROPOSE	1
				$ACCUSE	1	$PROTEST	1
						$FIGHT	1

Table 7.2

This table shows how individual sketchy scripts performed. The "WRONG BINDING" column indicates stories for which FRUMP selected the correct sketchy script but misunderstood information in the article. The "IGNORED" column contains stories that FRUMP did not process in spite of having the correct sketchy script. The "FALSE ALARM" column contains stories that FRUMP thought it understood but in fact did not.

CAUSE OF ERRORS

MISSING VOCABULARY	11
MISSING SCRIPT	9
COMPLEX SYNTAX	2
INCORRECTLY WRITTEN SCRIPT	1
PROBLEM RECOGNIZING NAME GROUP	1

Table 7.3

Table 7.3 shows the cause of the errors for the non-correct entries in table 7.2. The primary difficulty is vocabulary. The second most important reason for errors is lack of the correct script. Seven of the stories that appear in the "FALSE ALARM" category of table 7.2 would have been processed correctly if FRUMP had the appropriate sketchy script. For these, FRUMP's vocabulary would probably have to be augmented as well. Two stories were missed because FRUMP's knowledge of syntax was insufficient. One story was missed because the writer of the $MEET sketchy script neglected to include relevant information. Another story about a meeting was processed wrong because FRUMP did not correctly identify the members of a group of people.

SCRIPT SELECTION CONFUSION MATRIX

SCRIPT FRUMP SELECTED

CORRECT SCRIPT		A	B	C	D	E	F	G	H	I	TOTAL
A	$THREAT	2									2
B	$FIGHT		3							2	5
C	$COMMENT			3							3
D	$DEATH				2					2	4
E	$VEHAC					1				2	1
F	$MEET			1			2			2	5
G	$FIRE									4	4
H	OTHER			2						1	3
I	MISSING	1	1		1		2		4		9
	TOTAL	3	4	6	3	1	4	0	4	13	38
		A	B	C	D	E	F	G	H	I	

Table 7.4

Table 7.4 is a confusion matrix for script selection. Entries on the left side indicate the correct classification of stories. Thus there were 2 actual stories about one country threatening another, 5 stories about two countries fighting, etc. The columns indicate the way FRUMP classified the stories. FRUMP thought there were 3 threaten stories, 4 fighting stories, etc. The numbers along the principle diagonal are stories for which FRUMP selected the correct sketchy script.

7.4 FRUMP's Knowledge Base

FRUMP currently has 48 sketchy scripts in its repertoire. They are:

$ACCEPT-BID	$ACCUSE	$AGREE
$APPROVAL	$ARREST	$ASSAULT
$ASYLUM	$BLOCKADE	$BREAK-NEGOTIATIONS
$BREAK RELATIONS	$COMMENT	$DEATH
$DEMAND	$DEMONSTRATION	$DENY
$DEPORT	$DETAIN	$EARTHQUAKE
$ELECTION	$ESTABLISH-NEGOTIATIONS	$ESTABLISH-RELATIONS
$EXPEL	$EXPLOSION	$FIGHTING
$FIND-ASSET	$GIVE-AID	$HOSPITAL
$KIDNAP	$MEET	$MOBILIZATION
$NATIONALIZE	$NO-AGREEMENT	$POSTPONE
$PRICE-INCREASE	$PROPOSE	$PROPOSE2
$PROTEST	$REDUCE-AID	$REJECT
$SEIZURE	$SPILL	$STORM
$STRIKE	$STRUCTURE-FIRE	$TERROR
$THREATEN	$VEHICLE-ACCIDENT	$WEAPONS-TEST

FRUMP's present vocabulary is approximately 1100 words. This is relatively small and it accounts for the most important reason for missing stories in the April 5 run. As table 7.3 shows, 11 stories were missed or incorrectly processed due to insufficient vocabulary. The number of stories correctly processed would be doubled but for vocabulary problems.

There are no insurmountable problems associated with increasing FRUMP's vocabulary. As we saw in chapter 5 the dictionary definition of a single word is quite simple and therefore uses very little storage. Furthermore, FRUMP's efficiency is not compromised by increased vocabulary. None of FRUMP's processing is sensitive to the size of its vocabulary. Thus the largest obstacle to greater success for the system is the effort required to add four or five thousand more vocabulary items.

CHAPTER 8

EXTENDING THE PREDICTOR: ORGANIZING SKETCHY SCRIPTS

8.1 Introduction

Often understanding an article solely in terms of its relevant scripts is insufficient for producing an acceptable summary. Rather, the situations that make up the article must be related to each other and to scripty situations from previous articles. Consider the following first paragraphs of two New York Times articles:

1) The State Department released a statement today that certain economic sanctions against Cuba would be lifted - among them the 12-year old ban on exports to Cuba by foreign subsidiaries of American companies. However, the State Department spokesman said that the embargo on direct trade between Cuba and the United States remained in force.

2) In a two hour news conference Premier Fidel Castro tonight praised the decision of the United States to permit increased trade with Cuba.

The only sketchy script identifiable in the first article is the script to handle announcements of organizations, $ANNOUNCE. The important items in $ANNOUNCE are the identity of the organization making the announcement, the identity of the person representing the organization who acts as the spokesman, and what was announced. FRUMP understands this story as the United States State Department announcing improved economic relations with Cuba.

The appropriate sketchy script in the second story is $PRESS-CONFERENCE; the story is about a press conference situation. The press conference sketchy script tries to identify who is holding the conference and what he said. FRUMP's final representation of the second story is an MTRANS by Castro that he approves of an action by the United States, and that action is the improved economic relations with Cuba.

These two articles relate strongly to each other. Any person reading them both realizes that the action by the United States reported in the first article is the action that was praised by Cuba in the second. FRUMP must also connect the two instantiated sketchy scripts for these articles. Relating the representations of different news events has two advantages: 1) it allows FRUMP to construct a more complete and accurate summary and 2) it makes FRUMP's processing more efficient and robust.

The best summaries of later articles often use information from former articles. A good summary of the second article is "In a press conference Castro said he approved of the announcement of the U.S. State Department to relax economic sanctions against Cuba." This summary includes the information from the second article that Castro praised the U.S. and information from the first article that the action being praised is lifting economic sanctions by the State Department. In order to construct a summary such as this the two articles must somehow be related within FRUMP's memory. If they are not related, the summary of the second article can include only the less accurate phrase "improved economic relations" rather than the more explicit "reduced economic sanctions." The more explicit information exists only in the representation of the first article. Thus relating the articles in multi-article news events is important for summarization.

Connecting articles about related news topics in FRUMP's memory can also make FRUMP's processing more robust. Suppose that for some reason FRUMP were unable to understand the action for which Castro praised the U.S. in the second story. This could happen if FRUMP did not know one of the important words or phrases or if the article specified the action using an odd or complicated sentence structure or if an important input word were misspelled. If no attempt is made to connect it with the previous story, FRUMP will understand it only as "Castro praised the U.S." That is also the best summary that can be produced. If, however, FRUMP is able to connect the representation of the second story with the first via the inference that the action for which Castro praised the U.S. is probably the action announced by the State Department. Then the representation of the second story will be complete and the system will still be able to

provide the summary "In a press conference Castro said he approved of the announcement of the U.S. State Department to relax economic sanctions against Cuba." The required inference can be made by appealing to information about situations in which countries "praise" each other, and on information about embargoes.

Furthermore, if FRUMP is able to connect the instantiated sketchy script of a story to previous related articles before it has finished processing the story, the information from the previous articles can often be used to make processing the current story more efficient. For example, FRUMP is able to connect the story about Castro's press conference with the previously processed story about the State Department's announcement before it has discovered why Castro is praising the U.S. This enables FRUMP to make a very strong and explicit prediction about what the praising is likely to be about. As we saw in chapters 4 and 5 FRUMP's processing is most efficient when it is able to make very explicit predictions.

Thus FRUMP can benefit from connecting related news articles in its memory. Two problems remain: exactly what is the organizing structure, and how can FRUMP recognize that a particular news situation fits in one of these structures?

8.2 Issue Skeletons

A news issue is a large news event encompassing more than one script situation. For example, a disagreement between countries might result in dissolution of diplomatic ties, border incidents, and eventually an all-out war. Each of these items is a news situation in itself, and therefore, each has its own sketchy script. However, they must be considered together to be recognized as a reasonable way for a shooting war to start. Another example is a political campaign. It consists of an announcement of candidacy, a number of campaign activities such as giving speeches, a convention, more campaign activities, and an election. Each of these components are sketchy script situations as well. However, a larger news issue unfolds when they are related to each other. The events surrounding a crime comprise a news issue: the crime is committed, a suspect is arrested, and a trial is convened. Each of these is a sketchy script situation in its own right but together they form a news issue. Most scripty news articles report situations that are actually part of a larger news issue.

An issue skeleton is the data structure which specifies to FRUMP how a news issue normally progresses. It does this by specifying what script situations are likely, in what order to expect them, and how each follows from the last. Consider, for example, the natural disaster issue skeleton discussed briefly in chapter 4.

NATURAL DISASTER ISSUE SKELETON

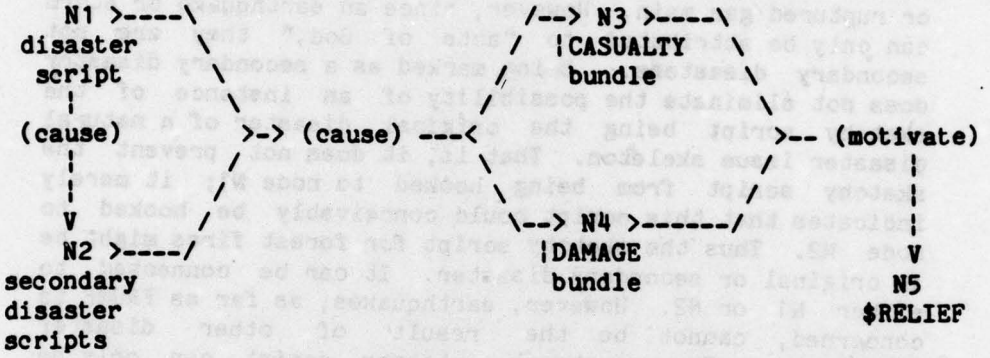


figure 8.1

Issue skeletons are made up of nodes and directed links between the nodes. In computer science terms it is an acyclic directed graph. In the natural disaster issue skeleton in figure 8.1 there are five such nodes. Each node of an issue skeleton can be attached to one or, in some cases, several instantiated sketchy scripts. At each node are constraints as to what sketchy script or class of sketchy scripts can be hooked to it. The links between the nodes indicate how the script situations at the different nodes are related. For example, nodes N1 and N2 are connected by a "cause" link which indicates an unknown causal relation. This link in figure 8.1 means that the original disaster can lead to other disasters through some perhaps complex and, to FRUMP, unanalyzed causal connection. Nodes N3 and N4 are connected by a "motivate" link to node N5. Nodes N3 and N4 represent the human casualties and property damage from the combined disasters. The link indicates that the casualties and property damage can motivate some other actor to give relief aid to the stricken area. That is, some other actor might instantiate the sketchy script $RELIEF with the recipient being the same location as in the disaster sketchy scripts.

Figure 8.1 says that a natural disaster news issue is (as far as FRUMP is concerned) an original disaster which can cause a number of secondary disasters, human casualties,

and property damage. Further, the casualties and property damage can then motivate relief efforts to the affected area.

As an aside, a secondary disaster is a class of sketchy script situations. A sketchy script for a disaster situation which can conceivably have a non-natural cause is labeled a secondary disaster sketchy script. This classification is necessary if FRUMP is to decide which is the original disaster. A forest fire is classified as a secondary disaster since it might be started by an arsonist or ruptured gas main. However, since an earthquake or storm can only be attributed to "acts of God," they are not secondary disasters. Being marked as a secondary disaster does not eliminate the possibility of an instance of the sketchy script being the original disaster of a natural disaster issue skeleton. That is, it does not prevent the sketchy script from being hooked to node N1; it merely indicates that this script could conceivably be hooked to node N2. Thus the sketchy script for forest fires might be an original or secondary disaster. It can be connected to either N1 or N2. However, earthquakes, as far as FRUMP is concerned, cannot be the result of other disaster situations. The earthquake sketchy script can only be connected to N1, not N2.

Although most of FRUMP's issue skeletons are similar to the natural disaster issue skeleton in complexity. However, some are very much more complex. The level of complexity of an issue skeleton is determined by how much we want FRUMP to find out about about instances of that news issue. If it is important for FRUMP to understand many details about a news issue and if those details are usually reported, the issue skeleton for that news issue will be complicated. If, on the other hand, it is not important or not possible for FRUMP to pick up many different facts about a news issue, the issue skeleton will be relatively simple.

8.3 What Makes Up an Issue Skeleton

There are two kinds of nodes and three kinds of links possible in an issue skeleton. These will be discussed next.

8.3.1 Kinds of Nodes

Nodes can be either single script instance nodes or multiple instance nodes. This simply indicates the number of different instantiated sketchy scripts that can be attached to the node. An example of a single instance node is N1 in figure 8.1. There can be only one initial natural disaster in the natural disaster issue skeleton. Thus node N1 might be connected to an instantiated storm sketchy script or an instantiated earthquake sketchy script or an instantiated forest fire sketchy script etc. But it can be only one.

Node N2, on the other hand, is a multiple instance node. Any number of instances of secondary disasters can be attached to that node. Thus a storm might cause instances of flooding and power outages and highway accidents etc. For each secondary disaster there is an instantiated sketchy script and they are all connected to N2.

All nodes, whether they are connected to single instances or multiple instances, have conditions on what kind of situation can be attached to them. These conditions consist of constraints on what kind of generic script can be attached and constraints on the script variables of the attached sketchy scripts.

8.3.2 Types of Links within Issue Skeletons

There are three types of links permitted between sketchy scripts in issue skeletons: "cause," "enable," and "motivate." Such a link indicates how one script situation follows from another.

There has been no attempt made to construct a detailed representation for all the ways scripts interact. Instead of a large number of precise links we have identified a few general ones. A detailed analysis of links between scripts is neither necessary nor desirable for a system like FRUMP. Since FRUMP skims stories rather than reading them in detail, it often misses less important details of stories. It is just these details that are required to differentiate among exact links. Therefore to consider how scripts can be connected in a detailed way is inappropriate.

The "cause" link indicates that one sketchy script situation somehow was responsible for a second by primarily physical means. For example, nodes N1 and N2 in figure 8.1 are connected by a "cause" link. This indicates that the secondary disaster situations at N2 follow from the script situation at N1. Of course, the actual causative relation

is very complex and depends heavily on the actual sketchy scripts involved. That is, the low-level events necessary for an earthquake to cause an avalanche are very different from the low level events by which a storm causes flooding. However, to the level of detail of FRUMP's understanding, fine grained links are unnecessary.

The second kind of link is the "enable" link. It indicates that one script situation provides the necessary preconditions for another one. For example, the hospitalization issue skeleton looks like this:

THE HOSPITALIZATION ISSUE SKELETON

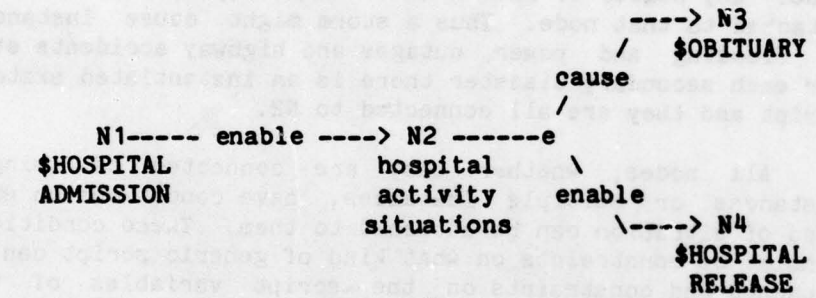


figure 8.2

Here N1 is a single instance node and N2 is a multiple instance node. "Hospital activity situations" is a class of sketchy scripts such as $SURGERY which are typically done at hospitals. Node N1 is connected to node N2 by an "enable" link because the most important function of the script situation at N1 is to provide a precondition for the situations at node N2. The "e" at the branch from N2 indicates that the links to N3 and N4 are exclusive. That is, either one can be taken but not both. This is different from the multiple links from nodes in the natural disaster issue skeleton. There the original disaster could possibly cause secondary disasters, casualties, and property damage.

The third kind of link is "motivate." It is used to indicate that a situation has resulted in a sentient actor initiating another situation. This is illustrated by the issue skeleton for labor negotiations. Here, a negotiation sketchy script is started over a labor dispute. The negotiations script can motivate a strike (if it ends unsuccessfully) or enable a vote on the new contract (if it is successful).

THE LABOR NEGOTIATIONS ISSUE SKELETON

]

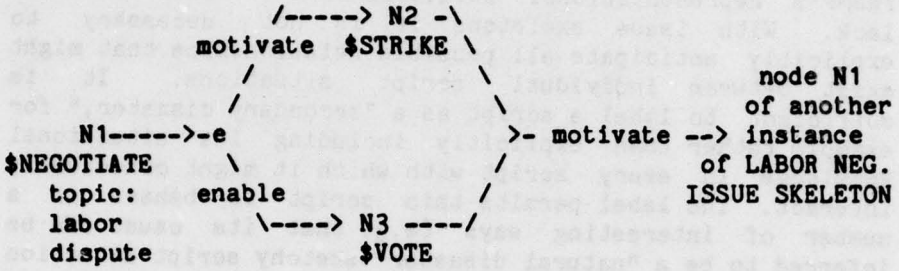


figure 8.3

If there is a strike it can in turn motivate the disputing parties to negotiate further, and the cycle can repeat. If the negotiations succeed, there can be a vote. If the vote is favorable, then the issue skeleton is complete. If the vote fails, it can motivate further negotiations. Again a detailed analysis of the relation between scripts connected with "motivate" is very complex. However, for FRUMP's purposes, "motivate" is sufficient.

8.4 Differences Between Issue Skeletons and Sketchy Scripts

By now it should be apparent that an issue skeleton's purpose is to impose a structure on collections of sketchy scripts. However, since structure is also permitted within a sketchy script (in terms of its tracks) this raises the question as to what determines that a certain structure is given by an issue skeleton rather than a sketchy script?

The answer is partly dictated by storage efficiency. Consider the ramifications of storing the $NATURAL-DISASTER structure at the script level. Every instance of a possible natural disaster (e.g. earthquake, flood, tornado, hurricane, etc.) would include the structure now in the natural disaster issue skeleton. That is, each would have predictions about relief efforts, casualty, and damage events. This information would be duplicated in the system once for every natural disaster situation.

More important to the distinction between issue skeletons and sketchy scripts is the fact that sketchy scripts represent small well defined specific situations while issue skeletons represent relations between these situations. Issue skeletons provide a generative nature to FRUMP's representational abilities that it would otherwise lack. With issue skeletons it is not necessary to explicitly anticipate all possible relationships that might exist between individual script situations. It is sufficient to label a script as a "secondary disaster," for example rather than explicitly including its situational knowledge in every script with which it might conceivably interact. The label permits this script to behave in a number of interesting ways (e.g. that its cause can be inferred to be a "natural disaster" sketchy script situation and that when summarizing either the causing disaster or secondary disaster, reference should be made to the other, etc). By using small sketchy scripts to represent situations, and issue skeletons to represent interactions between these sketchy scripts, it is possible for FRUMP to build large complicated representations out of individually simple pieces. Without something like issue skeletons this would not be possible. All of the possible interactions of each individual script with each other script would have to be explicitly stated.

8.5 Sketchy Script Constraints at Issue Skeleton Nodes

At each node in an issue skeleton there are constraints on what sketchy scripts can be used. These constraints dictate whether or not a particular instantiated sketchy script can be connected to a given issue skeleton. The constraints are of two types: static and dynamic.

Static constraints do not depend on the particular instances of sketchy scripts. Rather they constrain the kind of sketchy script that is acceptable at a node. For example, the labor negotiations issue skeleton (figure 8.3) only permits $STRIKE sketchy scripts to be hooked to node N2. An instance of a vehicle accident would not be acceptable. Thus the static constraints at N2 specify the generic sketchy scripts that will be permitted.

Dynamic constraints test script variable bindings. For example, node N1 in figure 8.3 represents negotiations between labor and management. It would not be acceptable to include a negotiation sketchy script in which the topic being negotiated was a peace treaty between two countries. Dynamic constraints eliminate this possibility by dictating acceptability requirements for script variable bindings. Dynamic constraints also are used to insure consistency

among instances of sketchy scripts in an issue skeleton. A new instance of $VOTE ought not be connected to node N3 if the union voting is different than the union participating in the negotiation of node N1. There is a dynamic constraint that tests whether the two unions are the same. If they are not, the $VOTE sketchy script is not attached to node N3. The new instance of $VOTE is connected only if there are no constraint violations.

Issue Skeleton constraints can be looked at as a kind of inference mechanism. If a newly instantiated sketchy script satisfies all of the constraints at a particular issue skeleton node, the situation represented by the new sketchy script is inferred to be part of the same news issue that the issue skeleton represents. For example, suppose a natural disaster issue skeleton has been built from one or more stories reporting a severe earthquake in Malaysia. The issue skeleton includes an earthquake sketchy script and a casualty bundle. Now suppose a new story is being read that FRUMP decides via one of its script selection procedures is an instance of $RELIEF, the sketchy script for relief efforts. If the relief efforts are directed to Malaysia and shortly after the earthquake, it is very likely that these efforts belong to the same news issue. The issue skeleton constraints provide FRUMP a mechanism for evaluating when to connect a new sketchy script to an existing issue skeleton. If all of the constraint tests are true, FRUMP infers that the sketchy script belongs in the issue skeleton.

8.6 Issue Skeletons that Share Sketchy Scripts

There are times when an instantiated sketchy script can be part of more than one news issue. In these cases the instantiated script must appear in multiple issue skeletons. The final conceptual representation is then several issue skeletons connected by a common instantiated sketchy script. For example, consider the following news story:

> A United Airlines jetliner en route to Miami was hijacked today by two gunmen who threatened to blow up the plane in flight. The gunmen, reportedly members of an underground organization seeking independence for Puerto Rico, ordered the release of three "political prisoners" being held in Florida jails and demanded to be taken to Cuba. However, on landing in Havana, they were arrested and charged with hijacking. Officials there said they would be prosecuted "to the fullest extent possible."

This story about a gunman hijacking a plane fits into both the crime issue skeleton and the terrorist issue skeleton. Both issue skeletons must be used if FRUMP is to understand all it can about the story. FRUMP must identify the crime issue skeleton if it is to understand the hijackers being arrested and charged. It must identify the terrorist issue skeleton in order to realize that demanding the release of prisoners is important. Thus issue skeletons can interact by sharing instantiated sketchy scripts. When they do, it is important for FRUMP to initiate all the appropriate issue skeletons if it is to correctly understand all of the important facts.

This has implications for the summarization of a news event as well. A complete summary of this internal structure requires summaries of both issue skeletons. A summary of the above article must contain that there was a hijacking, that the hijackers made demands to the U.S., that the plane landed in Cuba, and that the hijackers were arrested. Such a summary requires information from both of the issue skeletons: the demands of the hijackers can be processed only with the terrorist issue skeleton while the arrest can only be handled by the crime issue skeleton.

8.7 How Variable Element Stories Can Be Processed

In chapter 1 it was stated that certain types of stories could not be processed by scripts, but that one of these types, the variable element story, could be processed with the help of other data structures. The data structure necessary is the issue skeleton.

A variable element story is one in which even though the events are stylized, FRUMP requires predictions that cannot be anticipated by the sketchy script. For example, consider the following stories:

> President Carter today signed a mutual defense treaty with most of the NATO countries in which the European countries, notably West Germany, agree to assume a larger percentage of the maintenance costs.

and

> The United States Senate this morning after very little debate approved the NATO treaty, already signed by Jimmy Carter, that reduces the proportion of NATO's expenses contributed by the U.S.

The first story is about a substantive international agreement. FRUMP has a script for that. The sketchy script dictates that it is important to find the kind of treaty and with whom the treaty is made.The first story therefore can be processed with scripts.

The second story, however, is about an action by the Senate. This is also a well defined sketchy script situation. However, one of the important points that must be included in the final representation if FRUMP is to understand a story about Senate action is what was voted on. Here, the script is of no use. There is no useful characterization of what the Senate can and cannot vote on. There are simply too many possible topics. Therefore, PREDICT cannot anticipate the topic of the Senate action which might then be communicated to SUBSTANTIATE to aid in the process of text analysis.

The script alone cannot help PREDICT anticipate the topic of the Senate action but the context set up by the first story can. The issue skeleton for treaties looks like this:

ISSUE SKELETON FOR INTERNATIONAL AGREEMENTS

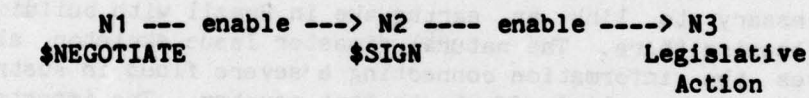


figure 8.4

Remember that there are constraints at the nodes of issue skeletons. One of the dynamic constraints on the sketchy script for the legislative action at N3 is that the topic voted on must be the treaty signed at N2. Since the script variable for the treaty is already bound by the first story to "a mutual defense treaty with NATO countries," this information is available during processing of the second story. Therefore, FRUMP can make a prediction about what the legislative action will be based on the issue skeleton. Of course, this may not be the only issue skeleton that requires a legislative action sketchy script. Suppose, for example, Carter had recently also signed a trade agreement with Japan. Therefore, FRUMP cannot know immediately which issue skeleton the current legislative action belongs with. However, this is exactly the problem that was addressed in sec 4.6 of chapter 4 in the discussion about predicting several explicit role fillers. In the case of the treaty stories, PREDICTOR tells SUBSTANTIATOR to add the topic of

the legislative action to the internal representation and predicts that it will be either an economic treaty with Japan or a defense treaty with NATO. Given this choice, it is not difficult for the text analyzer to realize that the treaty referred to by the phrase "the NATO treaty" is the latter alternative. Thus issue skeletons can provide certain predictive information of a kind that can inherently not be given by a script.

8.8 Conclusion

Issue skeletons are more than just a higher level of sketchy scripts. They provide for a general level of script interaction which would otherwise be missing. Each issue skeleton dictates in a general way how classes of sketchy scripts fit together. Thus issue skeletons permit FRUMP's sketchy scripts to be used as building blocks from which more complicated representational structures can be built. Issue skeletons can connect representational structures in two ways.

First, different stories can use the same type of issue skeleton to connect different sketchy scripts. For example, the natural disaster issue skeleton provides the information necessary to link an earthquake in Brazil with buildings collapsing there. The natural disaster issue skeleton also gives the information connecting a severe flood in Austria and Red Cross relief efforts to that country. The important point is that the information is specified in the issue skeleton at a general level. It can then apply equally well to any disaster situation.

The second way issue skeletons can tie representational structures together is by shared sketchy scripts. Several different instances of issue skeletons can incorporate the same sketchy script or request bundle. For example, imagine a story reporting that both an earthquake and a flood struck Italy. In that case there will probably be only one casualty figure. The article is unlikely to state how many people were killed due to the flood and how many due to the earthquake. Rather there will be only one number representing casualties from both disasters. Indeed, it is quite likely that the division between the disasters is impossible. In a case such as this, FRUMP will generate one issue skeleton for each disaster. However, the issue skeletons will share the same casualty bundle.

Thus issue skeletons can organize sketchy script interaction. This organization reflects the intrinsic organization of news issues. That is, the natural disaster issue skeleton relates disasters with relief efforts, for

example, because relief efforts are often related to real world disasters.

Because issue skeletons are still in the development stage, there are still problems to be worked out. The most important one is that there is currently no way to deactivate a viable issue skeleton. An issue skeleton must be kept around for a "reasonable" length of time after it is created in order to correctly process any articles that update its news issue. The problem is determining the time period after which it will be deactivated whether or not it has satisfied all of its nodes. If such deactivations are not allowed, the number of active issue skeletons might grow unboundedly. The effects of a flood (with potential secondary disasters such as cholera) may linger for weeks whereas a story describing an airplane crash will seldom be updated more than several days afterward. These problems will be delt with but as yet no solution has been implemented.

An issue skeleton is a useful data structure complementary to sketchy scripts. Issue skeletons can be used to connect in a coherent fashion the internal representations of related news stories. This is important not only for understanding but for generating summaries as well. Issue skeletons enable FRUMP to process a class of stories which would otherwise be missed.

CHAPTER 9

CONCLUSION

In this dissertation we explored the possibility of making pragmatic knowledge available to the text analyzer of a natural language processing system. The result is a system in which text analysis is highly integrated with the rest of the understanding process. We saw that an integrated system could indeed be constructed, and we demonstrated that the approach yields high returns in terms of processing efficiency and robustness.

The method of organizing such an integrated natural language processing system was outlined in chapter 1. Instead of the usual syntax, semantics, and pragmatic modules, the new organization is composed of a prediction module and a substantiation module. In this paradigm, the process of understanding natural language consists of generating hypotheses about what the text might mean and then finding a reading of the text that satisfies hypothesized constraints. The underlying virtue of this top down approach is that text analysis is almost always guided. Thus the text analyzer can be much simpler than in earlier systems. Initiating the top down understanding process was the subject of chapters 2 and 3 which discussed the problem of script selection.

A program (FRUMP) has been written using the new system organization. The system has access to the UPI news wire which supplies it with a constant stream of English text. Since FRUMP usually processes an average story in less than 20 seconds of CPU time on a DEC PDP 20/50, while UPI articles arrive at an average rate of one every 5 to 7 minutes, the system has no trouble staying ahead of the wire.

There are currently 48 sketchy scripts in FRUMP's repertoire. While still small compared to the number required to understand every scripty story on the UPI wire,

this does demonstrate that FRUMP is not inextricably tied to a particular micro-world. It also demonstrates that FRUMP can effectively choose among 48 scripts, at least. Furthermore, it was shown that the performance of the script selection algorithms is not significantly degraded by the addition of more scripts.

Other researchers have advocated text analysis using "meaning" information in addition to syntactic information (for example, Gershman [1979], Riesbeck & Schank [1976], Wilks [1973]). These systems, however, did not incorporate meaning predictions from pragmatic sources; they employed only semantic predictions.

SOPHIE, developed by Brown & Burton [1975], makes pragmatic information available to an integrated parser. However, the semantic grammar used by SOPHIE requires the input domain to be highly constrained. A semantic grammar is an augmented transition network except that the non-terminals are conceptual rather than syntactic items. Because the SOPHIE parser was integrated, it achieved both efficiency and robustness but at the expense of flexibility: SOPHIE's parser can only deal with inputs concerning trouble shooting a Heathkit IP-28 power supply. To add a new domain, a new semantic grammar must be written.

More recently, Carbonell [1979] added an integrated text analyzer to his POLITICS program. His system operates on several different domains concerning political events. The major difference between the POLITICS and FRUMP approach to integrated parsing is one of modularity. Carbonell argues against the division of natural language systems into a syntax module, a semantics module, and a pragmatics module. On that basis he advocates a completely non-modular system. In FRUMP, on the other hand, modularity plays a crucial role. FRUMP shares POLITIC's rejection of separate syntax, semantics, and pragmatics modules. However, rather than rejecting modularity altogether, FRUMP is an attempt to identify a more natural system decomposition. The FRUMP modules were described in detail in chapters 4 and 5. Modularization in FRUMP is important for two reasons. First, without it, managing a system the size of FRUMP would be nearly impossible. Second, it is difficult to discuss a system's contribution to a process model of understanding natural language without modularization.

There are two main limitations on FRUMP's language processing ability and two criticisms that might be made of the approach. We will now explore the validity of these and see how applicable they are to the FRUMP implementation and, more generally, to the FRUMP approach.

Limitation 1: FRUMP is only able to process script-based stories.

This limitation is traceable to FRUMP's PREDICTOR module. FRUMP uses a script applier as its predictive understander. Thus it is capable of processing only script-based stories. A script applier was chosen for FRUMP because it is better understood than other predictive understanders while still being applicable to an interesting set of input domains. While a script applier is an essential part of the implementation of FRUMP, the approach to integrated parsing does not require a script applier. No other predictive understander was used in FRUMP's implementation because no available version of other predictive understanders could produce the robust constraint predictions that the FRUMP approach requires.

Limitation 2: FRUMP is not capable of detailed understanding, only skimming.

Again this limitation is due to FRUMP's PREDICTOR module. FRUMP's processing is only as robust as the input constraints it generates. It is therefore imperative for the PREDICTOR to be robust. While a script applier is the most reliable and best understood prediction generator available, it is still too fragile to adequately test the idea of integrated text analysis. Any benefits of supplying pragmatic knowledge to the text analyzer are eliminated if the predictive understander is unable to supply pragmatic constraints. Therefore FRUMP uses a simplified script applier which makes fewer predictions but makes them consistently.

This limitation is also applicable only to the implementation of FRUMP, not the approach. When more sophisticated predictive understanders (e.g., Wilensky's Plan Applier [1978] and Carbonell's TRIAD [1979]) are extended to many more domains, they can be used to supplement the script applier's pragmatic constraints.

Criticism 1: FRUMP is too top down

The predominant processing in FRUMP is top down, but this an advantage rather than a disadvantage. For the most part FRUMP predicts only constraints, not explicit concepts. The point of integrated text analysis is that these top down constraints are nearly always available.

Furthermore, FRUMP is not always top down: selecting a sketchy script and initial analysis of a new phrase are often completely bottom up. The system does not make predictions but waits to see what the text analyzer can construct.

Criticism 2: FRUMP is not psychologically plausible.

The primary objection to FRUMP as a possible psychological model is the fact that its reading is not strictly left to right. Instead, it looks for a structure building word which is then used to guide further processing. However, there is some psychological evidence that fast human readers classify sentence types before analyzing the constituent words (Bower [1970]). Furthermore, there is ample evidence of retrograde eye movements in humans during skimming (Cunitz & Steinman [1969] and Tinker [1958]) although not as much as FRUMP currently does. FRUMP can easily be made to eliminate most of its implausible scanning by giving it a short term memory in which to store recently-looked-at words. However, psychological validity was not a major consideration in FRUMP's design. In any case, AI programs are interesting from a psychological point of view only when psychological analogies naturally fall out, not when they are built in.

Because FRUMP is robust, its performance on real world input can be evaluated. FRUMP typically understands between 8% and 12% of the stories from the UPI wire, although on an anomalous news day it can do much better or much worse. Since only about 50% of the UPI stories are understandable using the paradigm of a script applier, FRUMP is running at about 20% of the theoretical limit. The major reasons for missing the remaining scripty stories are: 1) lack of the correct sketchy script, 2) missing vocabulary words, and 3) use of unfamiliar sentence structures. Fixing the more important problems, the lack of scripts and vocabulary, poses no theoretical problems. While sentence structure problems might be theoretical, in the past they have not caused major rewrites in FRUMP. Problems thus far have been solved by modest improvements to FRUMP's syntactic heuristics. Thus the FRUMP experiment has been a success. The program is more robust on a wider range of input domains than any previous natural language system.

APPENDIX

THE INTERNAL VERSION OF THE FIGHTING SKETCHY SCRIPT

```
[
(CLASS SCRIPT)
(SVARIABLES &SIDE0 &SIDE1 &LOC &PROP)
(BUNDLE |CASUALTY |DAMAGE)
(SINITIATORS R0 R1 R2 R3 R4)
(ACTIVE R0 R1 R2 R3 R4 R5 R6 R7 R8)
]

[R0
 ((<=>($FIGHTING SIDE0 &SIDE0 SIDE1 &SIDE1)) LOC &LOC)

(SINIT
  ((<=> SIDE0) *POLITY*)
  ((<=> SIDE1) *POLITY*)
  ((LOC) NIL)
)

(SVARS
   (&SIDE0 (<=> SIDE0) *POLITY*)
   (&SIDE1 (<=> SIDE1) *POLITY*)
   (&LOC (LOC) *LOCATION*)
)
(CONSTRAINTS
   (DIFFERENT (<=> SIDE1) (<=> SIDE0))
)
]

[R1
 ((ACTOR &SIDE0 <=> (*PTRANS*) OBJECT (*TROOPS*) TO &SIDE1 ))

(SINIT
 ((ACTOR) *POLITY*)
 ((TO) *POLITY*)
)

(SVARS
 (&SIDE0 (ACTOR)  *POLITY*)
 (&SIDE1 (TO) *POLITY*)
)
(CONSTRAINTS
   (DIFFERENT (ACTOR) (TO))
)
(CAUSALS
 (INFER R0)
 (CANCAUSE R6 R7)
)
]
```

```
[R2
 ((ACTOR &SIDE1 <=> (*PTRANS*) OBJECT (*TROOPS*) TO &SIDE0))

(SINIT
 ((ACTOR) *POLITY*)
 ((TO) *POLITY*)
)

(SVARS
 (&SIDE1 (ACTOR)  *POLITY*)
 (&SIDE0 (TO) *POLITY*)
)
(CONSTRAINTS
   (DIFFERENT (ACTOR) (TO))
)
(CAUSALS
 (INFER R0)
 (CANCAUSE R5 R8)
)
]

[R3
 ((ACTOR &SIDE0  <=> (*PTRANS*) OBJECT (*BOMB*) TO &SIDE1))

(SINIT
 ((ACTOR PART) *POLITY*)
 ((TO) *POLITY*)
)
(SVARS
 (&SIDE0 (ACTOR PART)  *POLITY*)
 (&SIDE1 (TO) *POLITY*)
)
(CONSTRAINTS
   (DIFFERENT (ACTOR PART) (TO))
)
(CAUSALS
 (INFER R0)
 (CANCAUSE R6 R7)
)
]

[R4
 ((ACTOR &SIDE1 <=> (*PTRANS*) OBJECT (*BOMB*) TO &SIDE0))

(SINIT
 ((ACTOR PART) *POLITY*)
 ((TO) *POLITY*)
)
(SVARS
 (&SIDE1 (ACTOR PART)  *POLITY*)
 (&SIDE0 (TO) *POLITY*)
)
```

```
(CONSTRAINTS
   (DIFFERENT (ACTOR PART) (TO))
)
(CAUSALS
 (INFER R0)
 (CANCAUSE R5 R8)
)
]

[R5
 ((ACTOR &SIDE0  <=> (*MTRANS*) MOBJECT (*CONCEPT*
   TYPE (*SURRENDER*)) TO &SIDE1))
(SINIT
 ((ACTOR) *POLITY*)
 ((TO) NIL)
)
(SVARS
 (&SIDE0 (ACTOR) *POLITY*)
 (&SIDE1 (TO) *POLITY*)
)
(CONSTRAINTS
   (DIFFERENT (ACTOR) (TO))
)
]
[R6
 ((ACTOR &SIDE1  <=> (*MTRANS*) MOBJECT (*CONCEPT*
   TYPE (*SURRENDER*)) TO &SIDE0))
(SINIT
 ((ACTOR) *POLITY*)
 ((TO) NIL)
)
(SVARS
 (&SIDE1 (ACTOR) *POLITY*)
 (&SIDE0 (TO) *POLITY*)
)
(CONSTRAINTS
   (DIFFERENT (ACTOR) (TO))
)
]

[R7
 ((CON ((ACTOR &SIDE0 <=> (*DO*)))
   CAUSE
       ((ACTOR (&PROP PART &SIDE1) IS (*PSTATE* VAL (-10))))
  ))

(SINIT
  ((CON ACTOR) *POLITY*)
  ((CAUSE ACTOR) *MILITARY-VALUE*)
  ((CAUSE ACTOR PART) *POLITY*)
)

(SVARS
```

```
  (&SIDE0 (CON ACTOR) *POLITY*)
  (&PROP (CAUSE ACTOR) *MILITARY-VALUE*)
  (&SIDE1 (CAUSE ACTOR PART) *POLITY*)
 )
 ]

[R8
 ((CON ((ACTOR &SIDE1 <=> (*DO*)))
   CAUSE
       ((ACTOR (&PROP PART &SIDE0) IS (*PSTATE* VAL (-10))))
  ))

(SINIT
  ((CON ACTOR) *POLITY*)
  ((CAUSE ACTOR) *MILITARY-VALUE*)
  ((CAUSE ACTOR PART) *POLITY*)
)

(SVARS
  (&SIDE1 (CON ACTOR) *POLITY*)
  (&PROP (CAUSE ACTOR) *MILITARY-VALUE*)
  (&SIDE0 (CAUSE ACTOR PART) *POLITY*)
 )
 ]
```

The top line indicates that this structure is a script rather than a request bundle or issue skeleton. The second S-expression gives the script variables that appear in this script. The third list are the request bundles that can appear with this sketchy script. The last two lines of the heading are respectively the list of key requests, that is, the conceptualizations allowed to trigger the script, and the number of initially active requests. This script is made up of nine requests (R0 - R8). Each has an SINIT, an SVARS, and a CONSTRAINTS property which dictate matching requirements for the role fillers. These requirements are the basis of the predicted constraints made by the PREDICTOR during understanding.

BIBLIOGRAPHY

1. Bobrow, D. and Collins, A., (1975). Representation and Understanding, Academic Press, New York.

2. Bobrow, D. G., Kaplan, R. M., Kay, M., Norman, D. A., Thompson, H., and Winograd, T. (1977). GUS, a frame driven dialog system. Artificial Intelligence, Vol. 8, No. 1.

3. Bower, T. G. R. (1970). Reading by eye. In H. Levin and J. Williams (Eds.), Basic Studies on Reading, Basic Books Inc., New York.

4. Bransford, J. and Franks, J. (1971). The abstraction of linguistic ideas. Cognitive Psychology, Vol. 2, pp. 331-350.

5. Brown, J. S., and Burton, R. R. (1975). Multiple representations of knowledge for tutorial reasoning. In D. Bobrow and A. Collins (Eds.), Representation and Understanding, Academic Press, New York.

6. Burton, R. R. (1976). Semantic grammar: an engineering technique for constructing natural language understanding systems. BBN Report 3453, ICAI Report 3. Bolt Beranek and Newman Inc., Boston.

7. Charniak, E. (1972). Toward a model of childrens story comprehension. AITR-266, Artificial Intelligence Laboratory, MIT, Cambridge, MA.

8. Charniak, E. and Wilks, Y. (1976). Computational Semantics, North-Holland, Amsterdam.

9. Charniak, E. (1977). A framed PAINTING: the representation of a common sense knowledge fragment. Cognitive Science, Vol. 1 No. 4

10. Charniak, E. (1978). With a spoon in hand this must be the eating frame. Theoretical Issues in Natural Language Processing - 2. D. Waltz, General Chairman. University of Illinois, Urbana, IL.

11. Cullingford, R. (1978). Script application: computer understanding of newspaper stories. Ph.D. Thesis, Yale University, New Haven, CT. Computer Science Department Research Report 116.

12. Cunitz, R. and Steinman, R. (1969). Comparisons of saccadic eye movements during fixation and reading. Vision Research 9, pp. 683-693.

13. DeJong, G. (1977). Skimming newspaper stories by computer. Yale Computer Science Department Research Report 104, New Haven, CT.

14. Eisenstadt, M. (1976). Processing newspaper stories: some thoughts on fighting and stylistics. Proceedings of the Second AISB Summer Conference, Edinburgh.

15. Gershman, A. (1979). Knowledge-based parsing. Ph.D. Thesis, Yale University, New Haven, CT.

16. Goldstein, I. and Roberts, R. B. (1977). NUDGE, a knowledge-based scheduling program. Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, MA.

17. Heidorn, G .E. (1975). Augmented phrase structure grammars. Theoretical Issues in Natural Language Processing Workshop. R. Schank and B. Nash-Weber, General Chairmen. Cambridge, MA.

18. Kaplan, R. m. (1971). Augmented transition networks as psychological models of sentence comprehension. Proceedings of the Second International Joint Conference on Artificial Intelligence, London.

19. Lehnert, W. (unpublished). Script selection. Unpublished manuscript.

20. Lehnert, W. (1978). Representing physical objects in memory. Yale Computer Science Department Research Report 131. New Haven, CT.

21. Lehnert, W. and Burstein, M. (1979). The role of object primitives in natural language processing. Submitted to the Sixth International Joint Conference on Artificial Intelligence. Tokyo.

22. Lesser, R., Fennell, R., Erman, L., and Reddy, D. R. (1974). Organization of the HEARSAY II speech understanding system. IEEE Symposium on Speech Recognition. L. Erman (Ed.), Carnegie-Mellon University, Pittsburgh.

23. Lowerre, B. (1976). The HARPY speech recognition system. Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh.

24. Marcus, M. (1977). A theory of syntactic recognition for natural language. Ph.D. Thesis, MIT, Cambridge, MA.

25. McClelland, Charles A. (1969). International interaction analyses in the predictive mode. World Event/Interaction Survey Technical Report 3. University of Southern California, Los Angles.

26. Minsky, M. (1968). Semantic Information Processing, MIT Press, Cambridge, MA.

27. Minsky, M. (1975). A framework for representing knowledge. In P. Winston, (Ed.), The Psychology of Computer Vision, McGraw-Hill, New York.

28. Newell, A. (1973). Artificial intelligence and the concept of mind. In R. C. Schank and K. M. Colby (Eds.),Computer Models of Thought and Language, W. H. Freeman and Co., San Francisco.

29. Norman, D. A. and Bobrow, D. G. (1975). On data-limited and resource-limited processes. Cognitive Psychology, Vol. 7, pp. 44-64.

30. Parkinson, R., Colby, M., and Faught, W. (1976). Conversational language comprehension using integrated pattern-matching and limited parsing. Technical Report, UCLA Psychiatry Department, Los Angles, CA.

31. Quillian, M. R. (1968). Semantic memory. In M. Minsky (Ed.), Semantic Information Processing, MIT Press, Cambridge, MA.

32. Raphael, B. (1968). SIR: semantic information retrieval. In M. Minsky (Ed.), Semantic Information Processing, MIT Press, Cambridge, MA.

33. Reddy, D. R., Erman, R., Fennel, R., and Neely, R. (1973). The HEARSAY speech understanding system: an example of the recognition process. Third International Joint Conference on Artificial Intelligence, Stanford, CA.

34. Rieger, C. (1975). Conceptual memory. In R. Schank (Ed.), Conceptual Information Processing. North Holland, Amsterdam.

35. Riesbeck, C. K. (1975). Conceptual analysis. In R. Schank (Ed.), Conceptual Information Processing. North Holland, Amsterdam.

36. Riesbeck, C. K. and Schank, R. C. (1976). Comprehension by computer: expectation-based analysis of sentences in context. Yale Computer Science Department Research Report 78, New Haven, CT.

37. Rumelhart, D. (1975). Notes on a schema for stories. In D. Bobrow and A. Collins (Eds.), Representation and Understanding. Academic Press, New York.

38. Schank, R. C. (1972). Conceptual dependency: a theory of natural language understanding. Cognitive Psychology, 3, pp. 552-631.

39. Schank, R. C. and Colby, K. M. (1973) Computer Models of Thought and Language. W. H. Freeman and Co., San Francisco.

40. Schank, R. C. (1975A). Conceptual Information Processing. North Holland, Amsterdam.

41. Schank, R. C. and Yale A. I. Project (1975B). SAM: a story understander. Yale Computer Science Department Research Report 43, New Haven CT.

42. Schank, R. C. and Abelson, R. P. (1977). Scripts, Plans, Goals, and Understanding. Lawrence Erlbaum Press, Hillsdale, NJ.

43. Schank, R. and DeJong, G. (in press) Purposive understanding. In B. Meltzer and D. Michie (Eds.), Machine Intelligence 9, Edinburgh University Press, Edinburgh.

44. Scragg, G. (1976). Semantic nets as memory models. In E. Charniak and Y. Wilks (Eds.), Computational Semantics, North-Holland, Amsterdam.

45. Simmons, R. F. (1973). Semantic networks: their computation and use for understanding English sentences. In R. C. Schank and K. M. Colby (Eds.),Computer Models of Thought and Language, W. H. Freeman and Co., San Francisco.

46. Tinker, M. A. (1958). Recent studies of eye movements in reading. Psychological Bulletin 55, pp. 300-307.

47. Wilks, Y. (1973). An artificial intelligence approach to machine translation. In R. C. Schank and K. Colby (Eds.), Computer Models of Thought and Language. W. H. Freeman and Co., San Francisco.

48. Wilensky, R. (1978). Understanding goal-based stories. Ph.D. Thesis, Yale University, New Haven, CT. Computer Science Department Research Report 140.

49. Winograd, T. (1972). Understanding Natural Language. Academic Press, New York.

50. Winston, P. (1975). The Psychology of Computer Vision, McGraw-Hill, New York.

51. Woods, W. A. and Kaplan, R. M. (1971). The lunar sciences natural language information system. BBN Report 2265. Bolt Beranek and Newman Inc. Cambridge, MA.

52. Woods, W. A. and Nash-Webber, B. (1972). The lunar sciences natural language information system: final report. BBN Report 2378. Bolt Beranek and Newman Inc. Cambridge, MA.